Technical Report

IBM Software Solutions, Research Triangle Park, North Carolina, USA

How to Move a VisualAge TeamConnection Version 3 Database

Kevin Postreich, Evan Thompson, Lee Perlov

TR 29.3088 December 1999

TeamConnection Development
IBM Software Solutions
Research Triangle Park, North Carolina

ABSTRACT

This technical report provides information and procedures for moving a VisualAge TeamConnection V3 database.
The objective is to provide a guideline that will reduce errors and save time by avoiding common mistakes and procedural problems.

ITIRC KEYWORDS

VisualAge
TeamConnection
Db2
db2move
Import/Export

ABOUT THE AUTHORS

Kevin Postreich
Mr. Postreich is a staff software engineer with the VisualAge TeamConnection development group. He joined IBM in 1980 as an electronic engineer in Charlotte North Carolina.
He relocated to RTP as an MVS systems programmer.

Mr. Postreich is currently a member of the TeamConnection development/test team, and a member of the first defense team for customer support.

Evan Thompson
Mr. Thompson is an advisory software engineer with the VisualAge TeamConnection development team. Previously, Mr. Thompson managed the AD Solutions Performance team and held a variety of architecture, advanced design and development roles in Application Development and Distributed Systems groups in the Santa Teresa, Cary and Menlo Park programming labs.

Mr. Thompson is currently the team lead of performance tuning for VisualAge TeamConnection.

Lee R. Perlov
Mr. Perlov is a staff software engineer in the VisualAge TeamConnection/CMVC development team. He started working for IBM in 1985 in Gaithersburg, Md, working in the Federal Systems Division on various projects for the United States intelligence community. He then moved to RTP to work on library development and support.

Mr. Perlov received a B.S. degree in Accounting from the University of Florida in 1983. He also completed two years of graduate work in the Department of Computer Science at the University of Florida.

Mr. Perlov is currently a member of the VisualAge TeamConnection Services team, as well as family, system and web administrator for the services TeamConnection family.

Technical Report

IBM Software Solutions, Research Triangle Park, North Carolina, USA

ABSTRACT 1
ITIRC KEYWORDS 1
ABOUT THE AUTHORS 2
Important Notice: 3
Introduction 4
Moving a TeamConnection Version 3 Database to another machine of LIKE Operating Systems 5
When the database name will remain the same: 5
When the database is going to be renamed: 5
Moving a TeamConnection Version 3 Database to another machine of UNLIKE Operating Systems 6
DB2move Utility Method: 6
db2move procedures: 6
Import/Export Method (Recommended) 7
Figure #1 (Export) 9
Figure #2 (Import) 10
Figure #3 11
Sample Shell scripts for Unix: 12
dbexport.ksh 12
dbimport.ksh 12
tcexport1.ksh (Export tables, except fhvbulkdata > 32700) 13
tcimport1.cmd (import tables, except fhvbulkdata > 32700) 14

Important Notices:

These database move instructions require the following:

1. **TeamConnection v303 or later installed.**
2. **The source and Target versions of TeamConnection are at the same fixpak and hotfix level.**
   Use the ‘tclevel’ command to determine the level of TEamConnection. If tclevel is not found, then you are not running TeamConnection fixpak 3.03 or later. First upgrade both Source and Target systems with TC 3.03 or later and Db2 5.2 fixpak 8.
3. **After moving a TeamConnection database from Intel to Unix, or from Unix to Intel, you must run a utility called ‘fixbulk.exe’ to convert the bulkdata to the correct format. Since this utility causes irreversable changes, we do not ship it with the packaged product. You must obtain the a copy of this utility at the following web site.**

ftp://ftp.software.ibm.com/ps/products/teamconnection/papers

Introduction

Moving a VisualAge TeamConnection version 3 database can be done in one of two ways.
The method chosen should be based on your decision of the operating system of the target database, as well as the name of the target database.
Simply moving a database from one location to another of like operating systems where the database name will not change, you should choose to use the db2 backup and restore functions.

However, moving a database to a different operating system, or moving the database specifying a new target database name, you should choose to use the db2 Export and Import functions.

Both of these methods are described below, as well as useful information on the db2 utility 'db2move'.

Technical Report

IBM Software Solutions, Research Triangle Park, North Carolina, USA

Moving a TeamConnection Version 3 Database to another machine of *LIKE* Operating Systems

When the database name will remain the same:

This is a straight forward and simple task primarily because the database name is the same. For this procedure to work, however, your userid's on both the source and target machines **Must** be the same.

1. Issue the db2 backup command on the source machine.
   ```
   db2 “backup database databasename to /home/backup/dbname”
   ```
2. Move the backup directory structure to the target machine
3. Issue the db2 restore command on the target machine
   ```
   db2 “restore database databasename from  /home/backup/dbname”
   ```

When the database is going to be renamed:

When moving a TeamConnection V3 database where the database name is going to be changed, the db2 backup and restore commands will not work. You must use one of the methods described for moving databases to different operating systems.

Technical Report

IBM Software Solutions, Research Triangle Park, North Carolina, USA

Moving a TeamConnection Version 3 Database to another machine of *UNLIKE* Operating Systems

There is only one basic method of moving a TeamConnection v3 database from one operating system to another. That is using Import/Export. However, we will discuss two procedures to complete this task. Using the db2move method will work, but is very restrictive. For this reason we recommend using the Export/Import method describe below.

DB2move Utility Method:

DB2 ships a utility called ‘db2move’ which can successfully move a TeamConnection V3 database. Db2move requires the userid performing the import function be the same as the userid that performing the export function.

The ‘db2move’ utility is documented in the DB2 Administrators Guide in the section labeled ‘moving databases’. Please refer to the DB2 Admin guide for complete details on using this utility.

db2move procedures:

db2move is really easy to use and will work only if the userid of the destination system is the same as the userid of the source system.

To get the syntax of the command, type db2move without any parameters at the command prompt.

This is the easiest way to use it:

Export: (Source System)

1. Create an empty directory and change to that directory
2. Type:

```
db2move export <originalfamilyname>
```

Note:DB2 will create a long list of files in this directory.

Import: (Destination System)

1. Use dbcreate to create an empty family which we will call <NewNameForFamily>
2. Create an empty directory and change to that directory
3. Copy (or FTP) those binary files created in the temporary directory on (Source System) to this temporary directory
4. Type:

```
db2move import <NewNameForFamily>
```

Import/Export Method (Recommended)

1. backup your database using the db2 backup command
2. db2 connect to databasename
3. Modify tcexport1.ksh with the path information where the .ixf and lobs should be exported to.
4. run tcexport1.ksh (d:\migrate\migcmvc\tools)
   This will do a db2 list tables command and generate a file 'exportscript.ksh' that contains all of the export commands to export tables.
   FHVBULKDATA for length(data) < 32700 is exported
   We will handle larger files seperately in th enext steps.
5. Determine if you have files over 32700, and if so, how many. These two queries are used to determine how to export bulkdata greater than 32700.
   db2 "select max(length(data)) from fhvbulkdata"
   If the result of this query is >32700, then proceed with the next bullet, and step 6. Otherwise go to step 7.
   db2 "select count(*) from fhvbulkdata where length >=32700"
6. **fhvbulkdata2** will be used is an ixf file with long blobs in the lob subdirectory.
   The following command is an example of how to export the buldata > 32700.
   db2 "export to ixf/FHVBULKDATA2.ixf of ixf lobs to ixf/lob/LOBFile/ f00,f01,f02,f03,f04,f05,f06,f07,f08,f09,f10,f11,f12,f13 modified by LOBsInFile select * from FHVBULKDATA where length(data) >= 32700"

   Note that 1000 files of each lobFile prefix will be created in the "lobs to" directory. So,
   db2 "select count(*) from fhvbulkdata where length(data) >= 32700"
   should be issued to determine the number of file prefixes which need to be specified ... any prefixes can be used. The above sample was used for a family with 13,768 instances of length(data) >= 32700. Since file deltas and compression is used for text parts, these are primarily segments of large binary files.
7. Execute exportscript.ksh to export the tables except the bulkdata > 32700.
8. Move the following file to the target system. We found it easier to zip all of the files and ftp the zip file. It also reduces the transfer time.

$PWD/ixf/*.ixf	(generated by the export)
$PWD/ixf/lobs/*	(be exporting bulkdata > 32700 in step 6)
$TC_DBPATH/cfgField/*	(.tbl, .fmt etc)
$TC_DBPATH/config/*	(userExit, security)

9. Modify tcimport1.cmd by editing the correct path information where the eport files are located on your system.
10. Execute tcimport1.cmd to generate a file "importscript.cmd' with all of the tables to import except fhvbulkdta > 32700

11. Create the new TeamConnection family using the fhcirt program

```
fhcirt -c TC_HOME\nls\cfg\*.ddl
```

    Where TC_HOME = location of Teamc root directory. Example: d:\teamc

12. Execute importscript.cmd to import the tables.

```
importscript.cmd 2>&1 | tee importscript.out
```

13. Verify the Import was successful

```
grep -i "rows were rejected"  tcimport.out
```

    Verify that '0' rows were rejected for all commands
    See sample output in figure #3

14. Backup your database
15. If there was bulkdata > 32700 exported to fhvbulkdata2.ixf, we will import it using the following command. Assumes the exported lobs were placed in d:\backup\ixf\lobs.
    db2 "import from fhvbulkdata2.ixf of ixf lobs from d:\backup\ixf\lobs\ modified lobsinfile commitcCount 4096 insert into FHVBULKDATA"
16. fhcirt TC_HOME\nls\cfg\*.dd* TC_HOME\nls\cfg\*.bnd
17. Execute analyzeReorg according to the instructions for your operating system
18. Make a backup of your new family
19. Start the new family server.
20. **Run the 'fixbulk.exe' utility** to convert the bulkdata to the correct format for the new operating system. You only need to do this if moving from Unix to Intel or Intel to Unix.

Obtain the tool at ftp://ftp.software.ibm.com/ps/products/teamconnection/papers

NOTE: Failure to complete this step will result in the inability to extract, checkout, or view part contents from TeamConnection.

Figure #1 (Export)

```
db2 "connect to kevinlp"
db2 "export to ADOBJECT.ixf of ixf select * from        KEVINLP.ADOBJECT"
db2 "export to ADVIEWROOT.ixf of ixf select * from      KEVINLP.ADVIEWROOT"
db2 "export to FHCACCESS.ixf of ixf select * from       KEVINLP.FHCACCESS"
db2 "export to FHCAPPROVAL.ixf of ixf select * from     KEVINLP.FHCAPPROVAL"
db2 "export to FHCAPPROVER.ixf of ixf select * from     KEVINLP.FHCAPPROVER"
db2 "export to FHCAUTHORITY.ixf of ixf select * from    KEVINLP.FHCAUTHORITY"
db2 "export to FHCBUILDER.ixf of ixf select * from      KEVINLP.FHCBUILDER"
db2 "export to FHCCFGCOMPROC.ixf of ixf select * from   KEVINLP.FHCCFGCOMPROC"
db2 "export to FHCCFGRELPROC.ixf of ixf select * from   KEVINLP.FHCCFGRELPROC"
db2 "export to FHCCHANGE.ixf of ixf select * from       KEVINLP.FHCCHANGE"
db2 "export to FHCCOMP.ixf of ixf select * from         KEVINLP.FHCCOMP"
db2 "export to FHCCOMPMEMBER.ixf of ixf select * from   KEVINLP.FHCCOMPMEMBER"
db2 "export to FHCCONFFIELDDATA.ixf of ixf select * from KEVINLP.FHCCONFFIELDDATA"
db2 "export to FHCCONFIG.ixf of ixf select * from       KEVINLP.FHCCONFIG"
db2 "export to FHCDEFECT.ixf of ixf select * from       KEVINLP.FHCDEFECT"
db2 "export to FHCENVIRONMENT.ixf of ixf select * from  KEVINLP.FHCENVIRONMENT"
db2 "export to FHCFILE.ixf of ixf select * from         KEVINLP.FHCFILE"
db2 "export to FHCFIX.ixf of ixf select * from          KEVINLP.FHCFIX"
db2 "export to FHCHOST.ixf of ixf select * from         KEVINLP.FHCHOST"
db2 "export to FHCINTEREST.ixf of ixf select * from     KEVINLP.FHCINTEREST"
db2 "export to FHCLEVEL.ixf of ixf select * from        KEVINLP.FHCLEVEL"
db2 "export to FHCLEVELMEMBER.ixf of ixf select * from  KEVINLP.FHCLEVELMEMBER"
db2 "export to FHCNOTE.ixf of ixf select * from         KEVINLP.FHCNOTE"
db2 "export to FHCNOTIFY.ixf of ixf select * from       KEVINLP.FHCNOTIFY"
db2 "export to FHCOVERSION.ixf of ixf select * from     KEVINLP.FHCOVERSION"
db2 "export to FHCPARSER.ixf of ixf select * from       KEVINLP.FHCPARSER"
db2 "export to FHCPATH.ixf of ixf select * from         KEVINLP.FHCPATH"
db2 "export to FHCRELEASE.ixf of ixf select * from      KEVINLP.FHCRELEASE"
db2 "export to FHCSEQUENCE.ixf of ixf select * from     KEVINLP.FHCSEQUENCE"
db2 "export to FHCTEST.ixf of ixf select * from         KEVINLP.FHCTEST"
db2 "export to FHCTRACK.ixf of ixf select * from        KEVINLP.FHCTRACK"
db2 "export to FHCUSER.ixf of ixf select * from         KEVINLP.FHCUSER"
db2 "export to FHCVERIFY.ixf of ixf select * from       KEVINLP.FHCVERIFY"
db2 "export to FHCVERSION.ixf of ixf select * from      KEVINLP.FHCVERSION"
db2 "export to FHCWORKAREA.ixf of ixf select * from     KEVINLP.FHCWORKAREA"
db2 "export to FHVBULK.ixf of ixf select * from         KEVINLP.FHVBULK"
db2 "export to FHVBULKDATA.ixf of ixf select * from     KEVINLP.FHVBULKDATA"
db2 "export to MANYPTRTABLE.ixf of ixf select * from    KEVINLP.MANYPTRTABLE"
db2 "export to NUVERSIONS.ixf of ixf select * from      KEVINLP.NUVERSIONS"
db2 "export to TCCOLLECTORCLASS.ixf of ixf select * from KEVINLP.TCCOLLECTORCLASS"
db2 "export to TCSOURCEOBJECT.ixf of ixf select * from  KEVINLP.TCSOURCEOBJECT"
db2 "export to TCVERSIONCLASS.ixf of ixf select * from  KEVINLP.TCVERSIONCLASS"
db2 "connect reset"
```

Figure #2 (Import)

```
db2 connect to tcfam
db2 import from ADOBJECT.ixf of ixf insert_update into         nobody.ADOBJECT
db2 import from ADVIEWROOT.ixf of ixf insert_update into       nobody.ADVIEWROOT
db2 import from FHCACCESS.ixf of ixf insert_update into        nobody.FHCACCESS
db2 import from FHCAPPROVAL.ixf of ixf insert_update into      nobody.FHCAPPROVAL
db2 import from FHCAPPROVER.ixf of ixf insert_update into      nobody.FHCAPPROVER
db2 import from FHCBUILDER.ixf of ixf insert_update into       nobody.FHCBUILDER
db2 import from FHCCFGCOMPROC.ixf of ixf insert_update into    nobody.FHCCFGCOMPROC
db2 import from FHCCFGRELPROC.ixf of ixf insert_update into    nobody.FHCCFGRELPROC
db2 import from FHCCHANGE.ixf of ixf insert_update into        nobody.FHCCHANGE
db2 import from FHCCOMP.ixf of ixf insert_update into          nobody.FHCCOMP
db2 import from FHCCOMPMEMBER.ixf of ixf insert_update into    nobody.FHCCOMPMEMBER
db2 import from FHCCONFFIELDDATA.ixf of ixf insert_update into nobody.FHCCONFFIELDDATA
db2 import from FHCCONFIG.ixf of ixf insert_update into        nobody.FHCCONFIG
db2 import from FHCDEFECT.ixf of ixf insert_update into        nobody.FHCDEFECT
db2 import from FHCENVIRONMENT.ixf of ixf insert_update into   nobody.FHCENVIRONMENT
db2 import from FHCFILE.ixf of ixf insert_update into          nobody.FHCFILE
db2 import from FHCFIX.ixf of ixf insert_update into           nobody.FHCFIX
db2 import from FHCHOST.ixf of ixf insert_update into          nobody.FHCHOST
db2 import from FHCINTEREST.ixf of ixf insert_update into      nobody.FHCINTEREST
db2 import from FHCLEVEL.ixf of ixf insert_update into         nobody.FHCLEVEL
db2 import from FHCLEVELMEMBER.ixf of ixf insert_update into   nobody.FHCLEVELMEMBER
db2 import from FHCNOTE.ixf of ixf insert_update into          nobody.FHCNOTE
db2 import from FHCNOTIFY.ixf of ixf insert_update into        nobody.FHCNOTIFY
db2 import from FHCOVERSION.ixf of ixf insert_update into      nobody.FHCOVERSION
db2 import from FHCPARSER.ixf of ixf insert_update into        nobody.FHCPARSER
db2 import from FHCPATH.ixf of ixf insert_update into          nobody.FHCPATH
db2 import from FHCRELEASE.ixf of ixf insert_update into       nobody.FHCRELEASE
db2 import from FHCSEQUENCE.ixf of ixf insert_update into      nobody.FHCSEQUENCE
db2 import from FHCTEST.ixf of ixf insert_update into          nobody.FHCTEST
db2 import from FHCTRACK.ixf of ixf insert_update into         nobody.FHCTRACK
db2 import from FHCUSER.ixf of ixf insert_update into          nobody.FHCUSER
db2 import from FHCVERIFY.ixf of ixf insert_update into        nobody.FHCVERIFY
db2 import from FHCVERSION.ixf of ixf insert_update into       nobody.FHCVERSION
db2 import from FHCWORKAREA.ixf of ixf insert_update into      nobody.FHCWORKAREA
db2 import from FHVBULK.ixf of ixf insert_update into          nobody.FHVBULK
db2 import from FHVBULKDATA.ixf of ixf insert_update into      nobody.FHVBULKDATA
db2 import from MANYPTRTABLE.ixf of ixf insert_update into     nobody.MANYPTRTABLE
db2 import from NUVERSIONS.ixf of ixf insert_update into       nobody.NUVERSIONS
db2 import from TCCOLLECTORCLASS.ixf of ixf insert_update into nobody.TCCOLLECTORCLASS
db2 import from TCSOURCEOBJECT.ixf of ixf insert_update into   nobody.TCSOURCEOBJECT
db2 import from TCVERSIONCLASS.ixf of ixf insert_update into   nobody.TCVERSIONCLASS
db2 import from FHCAUTHORITY.ixf of ixf insert_update into     nobody.FHCAUTHORITY
db2 connect reset
```

Figure #3

```
D:\TEAMC\tcfam>db2 connect to tcfam

   Database Connection Information

 Database product       = DB2/NT 5.0.0
 SQL authorization ID   = NOBODY
 Local database alias   = TCFAM

D:\TEAMC\tcfam>db2 import from ADOBJECT.ixf of ixf insert_update into         nobody.ADOBJECT
SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19980925", and time "165429".

SQL3050W  Conversions on the data will be made between the IXF file code page
"819" and the application code page "1252".

SQL3153N  The T record in the PC/IXF file has name "ADOBJECT.ixf", qualifier "
", and source "           ".

SQL3109N  The utility is beginning to load data from file "ADOBJECT.ixf".

SQL3110N  The utility has completed processing.  "6728" rows were read from
the input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "6728".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "6728" rows were processed from the input file.  "0" rows were
successfully inserted into the table.  "0" rows were rejected.

Number of rows read         = 6728
Number of rows skipped      = 0
Number of rows inserted     = 0
Number of rows updated      = 6728
Number of rows rejected     = 0
Number of rows committed    = 6728

.......................
```

Sample Shell scripts for Unix:

dbexport.ksh

```
#!/usr/bin/ksh
print Exporting Tables for $(id -un) using IXF format files
cd
exportDir=${1:-$HOME/exports}
print into the \'$exportDir\' directory
if [[ ! -d $exportDir ]]
then
   print Creating $exportDir directory
   mkdir $exportDir
fi
cd $exportDir
db2 "connect to $(id -un)"
db2 "list tables" | awk '($3=="T") {print $1}' |
while read table
do
  print Exporting ${table}.IXF to $PWD
  db2 "export to ${table}.IXF of IXF select * from $table"
done
db2 "connect reset"
print Exports complete
```

dbimport.ksh

```
#!/usr/bin/ksh
print Importing Tables for $(id -un) using IXF format files
cd
exportDir=${1:-$HOME/exports}
print from the \'$exportDir\' directory
if [[ ! -d $exportDir ]]
then
   print Error, $exportDir directory missing
   exit 1
fi
cd $exportDir
db2 "connect to $(id -un)"
db2 "list tables" | awk '($3=="T") {print $1}' |
while read table
do
  print Importing ${table}.IXF from $PWD
  db2 "import from ${table}.IXF of IXF REPLACE into $table"
  #INSERT_UPDATE or REPLACE_CREATE
done
db2 "connect reset"
print Imports complete
```

tcexport1.ksh (Export tables, except fhvbulkdata > 32700)

```
#!/usr/bin/ksh
print " Generate script to Export Tables for $(id -un) using IXF format files"
ixfdir=${HOME}/backup/ixf
lobdir=${HOME}/backup/ixf/bulkLOB

if [[ ! -d $ixfdir ]]
then
  mkdir $ixfdir
fi

if [[ ! -d $lobdir ]]
then
  mkdir $lobdir
fi

print "#ixfdir=$ixfdir" | tee  exportscript.ksh
print "#lobdir=$lobdir" | tee -a  exportscript.ksh

db2 "connect to $(id -un)"
print "db2 \"connect to $(id -un)\"" | tee  -a exportscript.ksh

db2 "list tables" | awk '($3=="T") {print $1}' |

while read table
do

if [[ $table = "FHVBULKDATA" ]]
then
print "db2 \"export to ${ixfdir}/${table}.IXF of IXF select * from $table where length(data) <
32700 \"" | tee -a  exportscript.ksh
else
print "db2 \"export to ${ixfdir}/${table}.IXF of IXF select * from $table\"" | tee -a
exportscript.ksh
fi
done
db2 "connect reset"
print "db2 \"connect reset\"" | tee -a exportscript.ksh

print "exportscript.ksh generated"
```

tcimport1.cmd (import tables, except fhvbulkdata > 32700)

```
/*rexx*/
/*requires mks toolkit*/
famname="testfam"

say "Generate script to Import Tables for" famname "using IXF format files"

ixfdir="c:\import\ixf"
lobdir="c:\import\ixf\bulkLOB"

origdir=directory()
scriptFile=origdir"\importscript.cmd"
say scriptFile

say "/*ixfdir=" ixfdir "*/"
say "/*lobdir=" lobdir "*/"
logrc=lineout(scriptFile,'/*ixfdir='ixfdir'*/',1)

logrc=lineout(scriptFile,'/*lobdir='lobdir'*/')

db2 "connect to" famname
logrc=lineout(scriptFile,'db2 connect to 'famname)

outfile="tables.out"
gettables="db2 ""list tables"" | awk '($3==""T"") {print $1}' >"outfile
gettables

do until lines(outfile)=0
  tablename=linein(outfile)
  say "table "tablename "being processed."
  logrc=lineout(scriptFile,'db2 "import from 'ixfdir'\'tablename'.ixf of ixf commitCount 4096
insert_update into 'tablename'"')
end

db2 "connect reset"
say "db2 connect reset"
logrc=lineout(scriptFile,'db2 connect reset')

say scriptFile "generated"
say "modify the file for bulkdata stuff"
```

Technical Report

IBM Software Solutions, Research Triangle Park, North Carolina, USA