TEMENOS
The software specialist for banking and finance

TAFJ-Standalone

R16

Amendment History:

Revision	Date Amended	Name	Description
1	1st April 2011	TAFJ team	Initial version
2	7st February 2012	H. Aubert	R12GA review
3	16st January 2013	H. Aubert	R13GA review
4	19th February 2014	R. Vincent	R14GA review
5	20th February 2014	JN.Charpin	Reject parameter – tafj-maven-plugin compilation
6	15th April 2014	H.AUBERT	R14GA review
7	6th May 2014	JN. Charpin	Logger context
8	10th March 2015	H. Aubert	R15 AMR review
9	19th March 2015	JN. Charpin	New compiler property
10	31th March 2015	JN. Charpin	Log context setup
11	15th March 2016	M.Kumar	R16 AMR review

Copyright

Errata and Comments

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

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

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

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

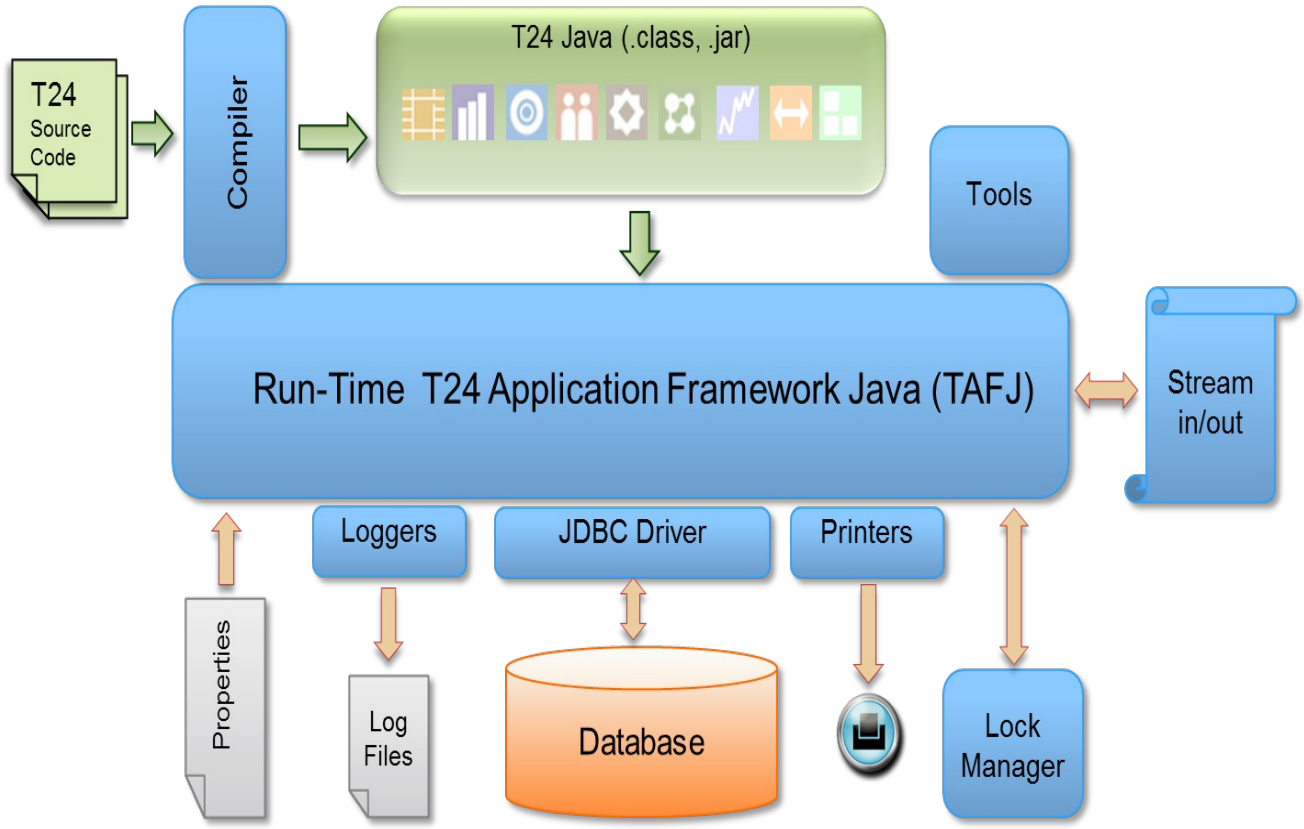
Table of Contents

What is TAFJ

Introduction

TAFJ (Temenos Application Framework Java) is a Pick BASIC runtime and compiler, written in 100% java. It allows compiling and running Pick BASIC programs. TAFJ has been written specifically for T24 needs, and all tests and proof of concepts have been done using T24. TAFJ also manage the connectivity on JDBC compliant databases like jBASE, Oracle, Microsoft SQL Server, DB2, derby, derby (Embedded). It comes with an embedded exporting tool for migrate your Pick data to Oracle.

Overview



Prerequisites

Java JDK

To install TAFJ, the only prerequisite is having a Java Development Kit (JDK) version 1.6.x or 1.7.x installed. To check what version of java you have, type in a console:

```
>java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

If the version reported is not 1.6.x or 1.7.x, you should install it. Java JDK can be downloaded here:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

NOTE: If you have installed java 1.6 but the command java -version doesn't report the correct version, this is certainly because your PATH environment variable points to another version. This is a bad practice to have your java virtual machine (VM) in the PATH. We highly recommend removing it. The best practice is to set-up an environment variable called JAVA_HOME pointing to the root directory of your Java Development Kit (JDK) installation. Then, you could reference JAVA_HOME/bin in your PATH environment variable.

The different scripts in ***<TAFJ_HOME>/bin*** will REQUIRE the environment variable ***JAVA_HOME***.

On Unix/Linux/:

```
export JAVA_HOME=<path_to_your_jdk>
```

Or update your file: ***.profile***

Multiple configuration

You can have multiple configuration file with one single setup of TAFJ. All these configuration files will have the extension **".properties"**. The default configuration is hidden in the file "***.default***" which resides in the conf directory. The Eclipse Builder plug-in will generate new configuration files using the Eclipse project name when you toggle a java project to a TAFJ project. You can also define new configurations manually. The default property file template is called "***.properties***".

The file ***.default*** in the ***<TAFJ_HOME>/conf*** directory can be used to set a default configuration. By default the value in the text file: ***.default*** is set to ***tafj.properties***. If no **-cf** argument is passed to tCompile, it will use the value in the file: ***.default*** (if the value in this file is a valid properties file).

So for our examples, we will imagine that we installed one TAFJ, but we have 2 releases of T24 : R14_3 and R14_5. For setting everything up from eclipse, please see the eclipse documentation. Here, we will do everything manually.

First, copy ***.properties*** in a file called R14_3.properties and a second time in a file called R14_5.properties.

In your conf directory, you should have:

.default
.properties
R14_3.properties
R14_5.properties

Then edit R14_3 and R14_5.properties.
By conventions, the java and class path for TAFJ of a project have to be set under:
<TAFJ_HOME>/data/<ProjectName>/java and classes
i.e
for R14_3.properties :

```
# Specify where the java files will be generated
# by the compiler
#
temn.tafj.directory.java      = <tafj.home>\data/R14_3/java

# Specify where the classes files will be generated
# by the compiler
#
temn.tafj.directory.classes     = <tafj.home>\data/myProject/classes
```

Once you are done with the modifications, you can go for a full compilation. See the compiler documentation for more information on compiling.

Example

Simple Program:

```
PROGRAM HELLO
   CRT "Sample program for TAFJCore"
   CRT "Please input your Name : " :
   INPUT NAME
   CRT "HELLO " : NAME
END
```

Save this program as HELLO.b

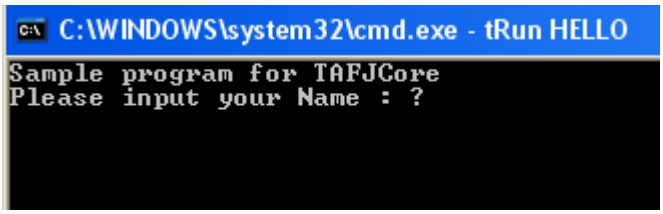
Compile this program using tCompile

> tCompile –cf R14_3 HELLO.b

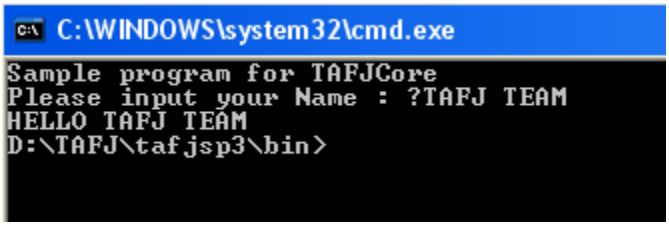
In order to run “HELLO” for R14_3, you will just type:

```
tRun -cf R14_3  HELLO
```

The Output will be:



After Name Entered:



To Compile the T24.BP for each R10 and R11 separately (T24.BP below is a directory of basic files).

> tCompile -cf R14_3 c:\product\T24\R10\T24.BP

> tCompile -cf R14_5 c:\product\T24\R11\T24.BP

In order to run a close of business for R10, one might start it with “START.TSM -DEBUG”, you will just type:

```
tRun -cf R10  START.TSM -DEBUG
```

If you want to do the same for R11, it will be:

```
tRun -cf R11  START.TSM -DEBUG
```

Default project

As previously mentioned, you can specify a default project in the file ***.default***. For example, with a fresh installation, the default configuration is tafj.properties, and thus the default file contains “tafj.properties”.

So in our previous example, if you write “R14_3” or “R14_3.properties” in the .default file, you won't need to specify the project when using tRun on the R10 configuration. You will then just have to type:

```
tRun  START.TSM -DEBUG
```

Logger

All executions are logged with TAFJ. TAFJ use the standard LOG4J open source.

The setting of the loggers for TAFJ is in the file ***<TAFJ_HOME>\conf\TAFJTrace.properties***.

In this file you can change the level and the appender of all loggers. For more information read the documentation of LOG4J.

http://logging.apache.org/log4j/1.2/manual.html

If the file doesn't exist, automatically TAFJ recreate a new one.

File TAFJTrace.properties :

```
#####################################
# Technology & Research Dep.
# Log file configuration
#
# TEMENOS (c) 2009
#
# This file contains configuration
# parameters for the log4j logger.
#
# Log Level = OFF, FATAL, ERROR, WARN, INFO, DEBUG
# Note :The 'File' must have only slashes '/' and never backslashes ''
#
#####################################
log4j.debug=false
log4j.rootLogger=OFF
log4j.logger.T24=INFO,       t24
log4j.logger.BASIC=ERROR,    basic
log4j.logger.PRINTER=ERROR,  printer
log4j.logger.DATABASE=ERROR, database
log4j.logger.JQL=ERROR,      jql
log4j.logger.LOCKING=INFO,   locking
log4j.logger.COMPILER=WARN,  compiler
log4j.logger.DEPENDENCY=INFO, dependency
log4j.logger.RUNTIME=ERROR,  runtime
log4j.logger.DBIMPORT=ERROR, dbimport
log4j.logger.SQLTRACE=ERROR, sqltrace
log4j.logger.ITYPE=ERROR,    itype
log4j.logger.EXECUTE=ERROR,  execute
log4j.logger.IOSERVER=INFO,  ioserver
log4j.logger.MDB=ERROR,      mdb
log4j.logger.EJB=ERROR,      ejb
log4j.logger.MONITOR=ERROR,  monitor
log4j.logger.FILTER=ERROR,   filter
log4j.logger.COBERTURA=ERROR,  cobertura
#log4j.logger.org.apache.tools.ant=ERROR, merge
log4j.logger.DBIMPORT-COMPARER=ERROR, dbcomparer
```

Log folders customization

By default you will find above mentioned logs under following folders:

- ***<TAFJ_HOME>/log : for TAFJ logs***
- ***<TAFJ_HOME>/log_T24 : for T24 logs***

This default behavior could be overridden by using following properties:

- **log.directory**=<PATH_TO_YOUR_TAFJ_LOG_FOLDER>
- **log.directory.t24**=<PATH_TO_YOUR_T24_LOG_FOLDER>

It has to be set at TAFJ start up, and should be done through environment variables or JVM arguments.

You cannot use the tafj.properties file for that purpose as loggers get initialized before tafj properties get loaded.

Set up through JVM argument (appserver mode):

```
-Dlog.directory=/path/to/your/custom/log/folder
```

Set up through environment variables (standalone mode tafj shell):

- Linux :
  - export log_directory='path/to/your/custom/log/directory'
- Windows
  - set log.directory=path\to\your\custom\log\directory

To revert this setting in tafj shell, simply set the property to an empty value.

- Linux :
  - export log_directory=
- Windows
  - set log.directory=

Logger context and multi-tenant.

In a multi-tenant environment you may want to define a specific logger context to differentiate the logs from the different tenants. It could also be useful for debugging purpose where a user can define its own context.

By using property

- **log.context**=<YOUR_CONTEXT_NAME>

you will find your logs under :

- ***<TAFJ_HOME>/log/<YOUR_CONTEXT_NAME> : for TAFJ logs***
- ***<TAFJ_HOME>/log_T24/<YOUR_CONTEXT_NAME> : for T24 logs***

Same feature apply to overridden log directory as mentioned above.

When using a context you will also generate a dedicated file to configure your appenders: TAFJTrace.<YOUR_CONTEXT_NAME>.properties file.

Like log directories properties it has to be set at TAFJ start up, and should be done through environment variables or JVM arguments, refer to log folder customization sample above.

Set up through JVM argument (appserver mode):

```
-Dlog.context=customLogContext
```

Set up through environment variables (standalone mode tafj shell):

- Linux :
  - export log_context=customLogContext
- Windows
  - set log.context=customLogContext

To revert this setting in tafj shell, simply set the property to an empty value.

- Linux :
  - export log_context=
- Windows
  - set log.context=

TAFJ on Multiple Servers

If you install TAFJ on multiple servers, you need then to setup the port range for each sever and be sure no port id will be duplicate on others server.

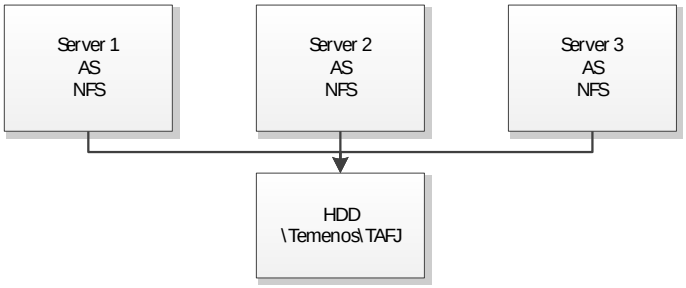
On each server in the properties file, set the key : `temn.tafj.runtime.port.range`

```
# set the port range of the system
#
#ie : temn.tafj.runtime.port.range = 10-45,76,89,2,130-150
temn.tafj.runtime.port.range                    =
```

Scenario 1

All the servers have a NFS/MAP to a share HDD where TAFJ is installed.

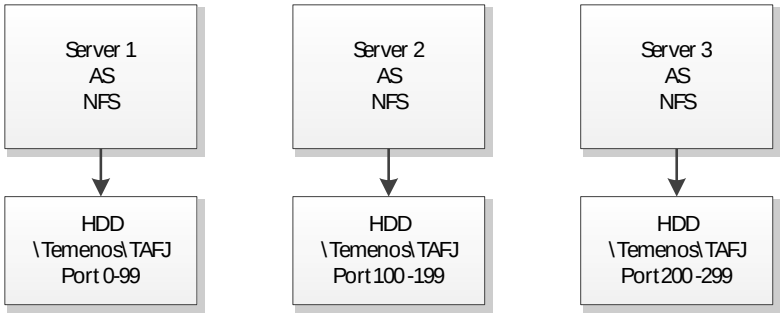
No Need to setup any port range. All servers will share the uid (unique ID file).



Scenario 2

Each servers have HDD file system where TAFJ is installed.

You need to setup for each TAFJ a port range. Each servers will have his own uid (unique ID file).



TAFJ-Compiler

Introduction

tCompile is the main entry point for compiling a program. This is a script in the /bin directory. It is used to execute the Compiled Basic Files

tCompile

The compilation of a basic program is done using tCompile (in the ***<TAFJ_HOME>/bin*** directory). As there is not a VOC as such, you must give the full path to the program you want to compile, unless the SOURCE is in the BASIC path specified in the tafj configuration file.

Syntax

The tCompile syntax is the following :

tCompile [-cf <confFile>] [-I <insertDir>] <program_path_and_name>

If the Directory is missing, TAFJ will try to find the file in the following order:

In the current directory
In the directories specified by "**temn.tafj.directory.basic**".

If the Directory is wrong, TAFJ will report an error.

Wild chars ('*'. '?') can't be used in the File name.

If the file is not found in the current directory, it will look at it in the path specified by "***temn.tafj.directory.basic***". However, if wild chars are used, only the current directory will be used.

If the -I option is passed, it will override the value of the "temn.tafj.directory.insert" property.

Examples

tCompile F.WRITE

Look for F.WRITE in the current directory, and if not found, uses the "temn.tafj.directory.basic"

tCompile /home/user/T24.BP/F.WRITE

Look for /home/user/GLOBUS.BP/F.WRITE and compile it.

tCompile /home/user/T24.BP

Compile the contents of the whole directory T24.BP.

tCompile /home/user/T24.BP/F.*

Compile only the file starting with "F."

tCompile ../T24.BP

Compile the entire relative directory.

Excluding files from compilation

When compiling a directory you could also exclude some files from the compilation by using the reject option.

```
tCompile [-cf <confFile>] [-reject rejectPattern] <program_path_and_name>
```

Examples

tCompile -reject */EB.* /home/user/T24.BP

Compile the content of the whole directory T24.BP but not the files starting with EB.

tCompile -reject */SUB/* /home/user/T24.BP

Compile the content of the whole directory T24.BP but not the subfolders called SUB.

tCompile -reject */SUB/EB.* /home/user/T24.BP

Compile the contents of the whole directory T24.BP but not the files starting with EB contained in subfolder SUB.

tCompile -reject “*/SUB/*|*/SUB2/*” /home/user/T24.BP

Compile the contents of the whole directory T24.BP but exclude subfolder SUB and SUB2.

You need to specify your reject pattern between double quotes when using multi-criteria pattern.

Sample basic file compilation:

```
--------------------------------------------------------------------------------
Temenos TAFJ Compiler/Runner
TAFJCompiler.jar version "DEV_201404.2"
Copyright (c) 2009 TEMENOS. All rights reserved
--------------------------------------------------------------------------------
Java Version = 1.7.0_51-b13
Java File Encoding = UTF-8
Java Home = D:\Temenos\T24Enterprise\3rdParty\java\jdk1.7.0_51-64\jre
Java Classpath = Java(TM) SE Runtime Environment
--------------------------------------------------------------------------------
User Name = haubert
Current Dir = D:\Temenos\T24Enterprise
--------------------------------------------------------------------------------
OS Type = Windows 7
--------------------------------------------------------------------------------
TAFJ_HOME   = D:\Temenos\T24Enterprise\TAFJ
Basic Source Dir        = D:\Temenos\T24Enterprise\TAFJ\samples\basic;
Insert Source Dir.      = D:\Temenos\T24Enterprise\TAFJ\samples\basic;
Precompile          Dir.=
Java Destination Dir.   = D:\Temenos\T24Enterprise\TAFJ\data/tafj/java
Classes Destination Dir.= D:\Temenos\T24Enterprise\TAFJ\data/tafj/classes
Report Destination Dir. = D:\Temenos\T24Enterprise\TAFJ\report
Max. Grammar Level      = 3
Min. Grammar Level      = 0
Java Package            = com.temenos.t24
Reporting               = false
--------------------------------------------------------------------------------
Arguments :HELLO
--------------------------------------------------------------------------------
Argument used : 'D:\Temenos\T24Enterprise\TAFJ\samples\basic\HELLO'

--------------------------------------------------------------------------------
T) Translate  C) Compile  G) Grammar
File(s) Basic                   File(s) Java               T C G  Rate   Status
--------------------------------------------------------------------------------
HELLO                           HELLO_cl                   V V 3[    -1][ DONE]
--------------------------------------------------------------------------------
Routine Fake because error
--------------------------------------------------------------------------------
Routine Fake because missing
--------------------------------------------------------------------------------
BASIC replacement
--------------------------------------------------------------------------------
Files requested  : 1
Files translated : 1
Files replaced   : 0
Files missing    : 0
Files trunced    : 0
Files compiled   : 1
Files canceled   : 0
Files with error : 0
--------------------------------------------------------------------------------
Total Time       : 0 [h] 0 [min] 2 [sec] 854 [ms]
================================================================================
```

Multiple basic file compilation:

tCompile FILE1 FILE2 FILE3 ...Up to...FILEN

```
--------------------------------------------------------------------------------
Temenos TAFJ Compiler/Runner
TAFJCompiler.jar version "DEV_201404.2"
Copyright (c) 2009 TEMENOS. All rights reserved
.
.
.
--------------------------------------------------------------------------------
Arguments :FILE1.b FILE2.b FILE3.b
--------------------------------------------------------------------------------
Argument used : 'D:\Temenos\T24Enterprise\TAFJ\samples\basic\FILE1.b'
Argument used : 'D:\Temenos\T24Enterprise\TAFJ\samples\basic\FILE2.b'
Argument used : 'D:\Temenos\T24Enterprise\TAFJ\samples\basic\FILE3.b'

--------------------------------------------------------------------------------
T) Translate  C) Compile  G) Grammar
File(s) Basic                   File(s) Java               T C G  Rate   Status
--------------------------------------------------------------------------------
FILE1                           FILE1_cl                   V V 3[    -1][ DONE]
FILE2                           FILE2_cl                   V V 3[    -1][ DONE]
FILE3                           FILE3_cl                   V V 3[    -1][ DONE]
--------------------------------------------------------------------------------
Routine Fake because error
--------------------------------------------------------------------------------
Routine Fake because missing
--------------------------------------------------------------------------------
BASIC replacement
--------------------------------------------------------------------------------
Files requested  : 3
Files translated : 3
Files replaced   : 0
Files missing    : 0
Files trunced    : 0
Files compiled   : 3
Files canceled   : 0
Files with error : 0
--------------------------------------------------------------------------------
Total Time       : 0 [h] 0 [min] 0 [sec] 818 [ms]
================================================================================
```

Understanding the compilation

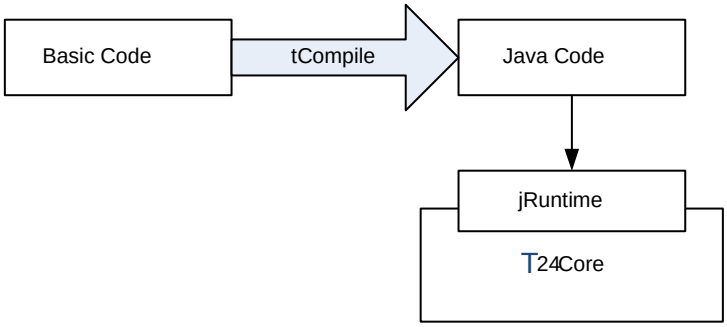
The compilation workflow

When a tCompile command is executed, first the BASIC file will be analyzed, so the grammar level can be defined. Then, the Java file will be generated **in memory**.

If the option **'temn.tafj.compiler.generate.class'** is set to true (default), the in-memory java file will be compiled and put in the directory specified by ***temn.tafj.directory.classes"***"

If the option **'temn.tafj.compiler.generate.java'** is set to true (default is 'false') , then the in-memory java file will be flushed on disk at the location specified by **'temn.tafj.directory.java'**

The generated java classes are all extending the main class jRuntime. This class (part of TAFJcore, contains the definitions of all the BASIC KEYWORDS (OPEN, OCONV, …)



The different grammars

During the compilation, you'll discover that there are different levels of so called grammars, depending on the BASIC file you are compiling.

These "grammars" are different ways of translating the BASIC to JAVA, depending on the quality of the BASIC CODE. Today, we have 3 main grammars levels: 2/3, 1 and 0

The grammar 2/3 implies that there are no GOTO, no ON … GOTO, no RETURN TO, and NO label in a "branch statement." A branch statement is the block of code you will find between :

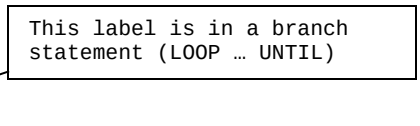
- IF … THEN … ELSE … END
- BEGIN CASE CASE … CASE … END CASE
- FOR … NEXT
- LOOP … WHILE | UNTIL test DO … REPEAT

The Grammar 2/3 is the most optimized, and the generated Java is well formed and more maintainable by a java developer.

The grammar 1 implies only no label in a branch statement. This grammar is not as optimized as the grammar 2/3, but can still be maintained.

The grammar 0 is due to the presence of a label in a branch statement.

For example, this will generate java in grammar 0 (Sample from BATCH.JOB.CONTROL):




```
    LOOP
       . . .
LOCK.CONTROL.LIST
        READU CONTROL.LIST FROM F.BATCH.STATUS, FLAG.ID LOCKED        ;* Get hold
of the control list
            GOTO LOCK.CONTROL.LIST
        END ELSE    ;* In case we are restarting - wait for lock!!!
END
    UNTIL CONTROL.LIST = PROCESSED      ;* Until everything has been done ; *
BG_100003752 S/E
       . . .
    REPEAT
```

The Label ‘LOCK.CONTROL.LIST ‘ is in a branch statement. Thus this program will compile in grammar 0 only. To correct it, a solution would be to modify it like this:

```
    LOOP
       . . .
LOOP
LOCK.CONTROL.LIST.AGAIN = 0
          READU CONTROL.LIST FROM F.BATCH.STATUS, FLAG.ID LOCKED        ;* Get
hold of the control list
              LOCK.CONTROL.LIST.AGAIN = 1
          END ELSE    ;* In case we are restarting - wait for lock!!!
             . . .
          END
WHILE LOCK.CONTROL.LIST.AGAIN = 1
REPEAT
    UNTIL CONTROL.LIST = PROCESSED      ;* Until everything has been done ; *
BG_100003752 S/E
       . . .
    REPEAT
```

The solution shown is not absolute. There are many ways to remove labels from a branch statement.

jBC source name

In general, only A..Z, a..z , 0..9 and '.' are accepted in PROGRAM / SUBROUTINE names. Few exceptions :

'$', '%' and '!' are also accepted, but there are exceptions and if possible, should not be used.

Note that the '-' is forbidden !

Java source name

The name of the java class is defined by the Basic code (SUBROUTINE, PROGRAM and FUNCTION), or if it doesn't exist, it is named by the name of the Basic file. Java doesn't support all the characters BASIC allows. The conversion rules for the names (Program, Subroutine, Function, VAR, Label, equate and insert are:

The '.' are replace by the '_'

The '%' are replace by the '_p_'

The '$' are replace by the '_d_'

The ' ' (space) are replace by the '_s_'

The '(' are replace by the '_l_'

The ')' are replace by the '_r_'

The '!' are ignored.

For example, file SLEEP.JBASE (SUBROUTINE !SLEEP$) is converted in

SLEEP_d_.java (SLEEP_d_.class)

BATCH.JOB.CONTROL is replace by BATCH_JOB_CONTROL

V$FUNCTION is replace by V_d_FUNCTION

Generation of files

As discussed, the compiler will generate two types of files. The java files containing the java source code and the class files containing the bytecode that can be executed.

You have to specify where the compiler has to generate these files.

With the key: "***temn.tafj.directory.java***", you specify where the java source files will be generate. This key is mandatory. And you can set only one path.

With the key: "***temn.tafj.directory.classes***", you specify where the classes files will be generate. This key is mandatory. And you can set only one path.

You have a special key: "***temn.tafj.directory.insert***". This key specifies where INSERT file is. The Insert file can be in the basic source folder or in the inserts paths.

Usually a java code is part of a package. The compiler creates a same package for all the code compiles. To set the package, use the key: "***temn.tafj.package"***.

Dependencies

There is no need to have all the SUBROUTINE to be able to compile a source.

Example :

```
PROGRAM A

    CALL B

END
```

You can tCompile A without having B. However, at runtime, if the class corresponding to “B” is not found, a runtime error will be raised.

Also, the tShow tool will report any missing dependency.

```
$ . ./tShow A

Home : '/home/user/tafj_dev/bin/..'


- Project : 'tafj'  [ FOUND ]

             BASIC source : '/home/user/tafj_dev/bin/./A.b'

            BASIC package : ''

          BASIC Import(s) : 'com.temenos.t24'

                                                              JAVA    class    :
'/home/user/tafj_dev/data/tafj/classes/com/temenos/t24/A_cl.class'

             Compiled the : 08 Oct 2015 09:09:52

                       on : chlap-user
      Compiled with TAFJ : DEV_201511

                Timestamp : 1444288192984
```

```
                  Grammar : 3

Include Basic Replacement : false



Checking dependencies ...

  [MISSING]

          Missing dependency : 'B' (com.temenos.t24.B_cl.class)

Check completed
```

Precompiled Basic Files

With the option "***temn.tafj.directory.precompile"***, you specify where the precompiled class files are.

You can specify multiple paths separate with the separator path. ":" or ";". These paths are overloaded by "***temn.tafj.directory.classes***" paths. If you have a class TEST.SUB in this path and compile a TEST.PRG call TEST.SUB. The TEST.PRG will call it. Now if you generate a new TEST.SUB in the "***temn.tafj.directory.classes***". The TEST.PRG will not call anymore the TEST.SUB in the "***temn.tafj.directory.precompile"*** but call the TEST.SUB in the "***temn.tafj.directory.classes".*** This path could be a folder or JAR file. This does not apply when running in an application server. In this case, the new class would have to be added to a jar within the application server's path.

Grammar Level

Already explained in the chapter "The different grammars" the level of the grammar is automatically defined by the compiler. You can force the maximum level and the minimum level of the grammar with the keys: "***temn.tafj.compiler.grammar.maxlevel"*** and "***temn.tafj.compiler.grammar.minlevel"***.

With the key ***"temn.tafj.compiler.grammar.minlevel"*** you can fix the minimum level of the grammar. If the compiler has to use a level smaller than the level you fix, then an error will be generated.

With the key "***temn.tafj.compiler.grammar.maxlevel"*** you can fix the maximum level of the grammar. It has to be bigger or equal of the minimum level grammar.

Compile for debugging

If you want to debug the basic process you have to compile with the option: "***temn.tafj.compiler.grammar.line.number"*** = true

This option will generate the java source code with lines numbers.

Warnings and errors

Each error is automatically shown in the console (depends the setting of the logger TAFJTrace.properties).

The compiler can log why a grammar level is downgraded with the key "***temn.tafj.compiler.grammar.warning.depreciation"*** and if an equate is overridden with the key ***temn.tafj.compiler.grammar.warning.equate***.

During the compilation of a Basic file, if a compilation error occurred, a java file, containing information to be printed on the screen when a call is executed, will be created.

In this case the compiler generate a **Fake because error** subroutine.

Example:

A program A

```
PROGRAM A
      CALL B
END
```

A subroutime B

```
SUBROUTINE B()
1   CRT "TOTO"
END
```

Error : `unexpected token: 1`

When you run A the output will be:

```
> Error on the file B
Line :3 -> unexpected token: 1
```

$INSERT Statement

By a T24 standard, a Basic Insert file should start with I_% like I_COMMON. With the key: "***temn.tafj.compiler.insert.name.any"*** set to true, the TAFJ compiler will accept any name as $INSERT file.

Basic file and Class file compliance

By default the basic source file encoding should be ISO-8859-1, but you may have to specify another file encoding when running on specific platform, i.e. ibm zOs.

”***temn.tafj.compiler.basic.encoding”*** = IBM-1047

A next section presents the way you could provide options to the java compiler.

A property allows you to define the java version compliance you want to use. i.e when setting 1.6 you will be able to execute the generated code on targeted version and later.

This property set both -source and -target javac arguments.

“temn.tafj.compiler.javac.compliance” = 1.6

JAVAC Options

The TAFJ Compiler uses javac to compile and generate class files. With the key: “***temn.tafj.compiler.javac.options”*** you can specify the standard javac options.

Examples on win32

```
Usage: javac <options> <source files>
where possible options include:
  -g                            Generate all debugging info
  -g:none                       Generate no debugging info
  -g:{lines,vars,source}        Generate only some debugging info
  -nowarn                       Generate no warnings
  -verbose                      Output messages about what the compiler is doing
  -deprecation                  Output source locations where deprecated APIs are used
  -classpath <path>             Specify where to find user class files and annotation processors
  -cp <path>                    Specify where to find user class files and annotation processors
  -sourcepath <path>            Specify where to find input source files
  -bootclasspath <path>         Override location of bootstrap class files
  -extdirs <dirs>               Override location of installed extensions
  -endorseddirs <dirs>          Override location of endorsed standards path
  -proc:{none,only}             Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...]   Names of the annotation processors to run; bypasses default
discovery process
  -processorpath <path>         Specify where to find annotation processors
  -d <directory>                Specify where to place generated class files
  -s <directory>                Specify where to place generated source files
  -implicit:{none,class}        Specify whether or not to generate class files for implicitly referenced
files
  -encoding <encoding>          Specify character encoding used by source files
  -source <release>             Provide source compatibility with specified release
  -target <release>             Generate class files for specific VM version
  -version                      Version information
  -help                         Print a synopsis of standard options
  -Akey[=value]                 Options to pass to annotation processors
  -X                            Print a synopsis of nonstandard options
  -J<flag>                      Pass <flag> directly to the runtime system
```

Other Compiler Properties

```
# When set to true enable compilation of new resources in eclipse
#
temn.tafj.compiler.eclipse.new.basic        = false

# When set to true enable the rating update in the BASIC code
#
temn.tafj.compiler.update.rating                    = false

# When set to true Eclipse compiler will consider only .b file
#
temn.tafj.compiler.eclipse.t24dev           = false

# When set to true stop the compilation if an error append and make a System.exit(-1)
# This property is not used by the Eclipse plug-in builder
temn.tafj.compiler.exit.if.error                    = false

# When compiling a .component file, where should we generate the
# different files and the header for TAFC ?
#
temn.tafj.compiler.tafc.component.dir               = <tafj.home>/tafc_components

# Eclipse basic editor. When set to true file parsing won't be processed on each CR
# To be used to speed up edition.
# Live code colorizer will be executed on save only.
#
temn.tafj.compiler.disable.cr.refresh = false
```

TAFJ maven plugin

TAFJ Basic compiler could be used through a maven build, which could be helpful if you want to integrate within a build or regression process a T24 compilation.

It will also speed up your compilation process depending on the compilation strategy you use.

If you are not familiar with the TAFJ maven plugin, first refer to the documentation “TAFJ MAVEN PLUGIN”. It will help you to get started with a simple demo.

This section assumes you are familiar with maven (commands, profile and lifecycle), the tafj maven project creation and the basic plugin usage.

It covers the various plugin options.

What we call the tafj maven project is the pom.xml which contains your build parameters.

You need to be connected to the temenos maven repository to use TAFJ maven capabilities.

T24 compilation

Common project setup

First create a pom.xml representing your tafj maven project, let's say BUILD_T24.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

      <modelVersion>4.0.0</modelVersion>

      <parent>
            <groupId>com.temenos.tafj</groupId>
            <artifactId>tafj-maven-parent</artifactId>
            <version>0.1.0</version>
      </parent>

      <groupId>com.temenos.t24</groupId>
      <name>BUILD_T24</name>
      <description></description>
      <artifactId>BUILD_T24</artifactId>
      <version>0.0.1-SNAPSHOT</version>
```

As explained in "TAFJ MAVEN PLUGIN" documentation, your project must inherit from latest `tafj-maven-parent` available.

We assume that you have a valid TAFJ installation under some path; a good practice is to define this path in the properties section of your project to be able to reuse it at multiple places.

```
<properties>
 <tafjHome>${basedir}/tafjHome</tafjHome>
 <lib.dir>${basedir}/lib</lib.dir>
 <project.build.sourceEncoding>cp1252</project.build.sourceEncoding>
</properties>
```

If the basic code you want to compile has some dependencies, for example T24 enterprise or components jar files, you should define the path to these jars in a property we call ${lib.dir} in same properties section.

Classic compilation

What we call the “classic compilation” is a compilation where tCompile is going to:

- Translate the basic files to java files
- Compile the java files to class files

The classic compilation is suitable for small projects, as it’s compiling file by file.

To illustrate this compilation, we define it within a profile called tcompile.

```
<profile>
 <id>tcompile</id>
 <build>
  <plugins>
   <plugin>
    <groupId>com.temenos.tafj</groupId>
    <artifactId>tafj-maven-plugin</artifactId>
    <executions>
     <execution>
      <phase>generate-sources</phase>
      <goals>
       <goal>compile</goal>
      </goals>
      <configuration>
       <tafjHome>${tafjHome}</tafjHome>
       <tafjProperties>tafj</tafjProperties>
       <basicDir>${basedir}/src/basic/BP</basicDir>
       <insertDir>${basedir}/src/basic/BP</insertDir>
       <javaDir>${basedir}/src/java</javaDir>
       <classesDir>${basedir}/target/classes</classesDir>
       <javaPackage>com.temenos.t24</javaPackage>
       <properties>
        <temn.tafj.compiler.generate.java>true</temn.tafj.compiler.generate.java>
        <temn.tafj.directory.precompile>${lib.dir</temn.tafj.directory.precompile>
       </properties>
      </configuration>
     </execution>
    </executions>
   </plugin>
  </plugins>
 </build>
</profile>
```

You can see that we refer to the plugin: tafj-maven-plugin and its goal compile.

By default, the goal compile is bound to the maven phase “generate-sources”, you could change this phase to fit your build requirement.

You need to configure the configuration section with your compilation parameters:

- The tafj home
- The properties file (tafj project) to use during compilation
- The basic folder to compile
- The insert directory, optional, could be read from the tafj properties file
- The java directory, optional, could be read from the tafj properties file
- The classes directory, optional, could be read from the tafj properties file
- The java package, optional, could be read from the tafj properties file
- Some additional specific tafj properties could be defined under a section <properties>

You have to define in this section: `<temn.tafj.directory.precompile>${lib.dir}</temn.tafj.directory.precompile>`
To resolve the dependencies you may have.

If you don't change de default phase binding you simply have to run from the root folder of your project command:

mvn clean generate-sources -P tcompile

You will have your basic files compiled to the classes directory you specified.

If you change the phase binding, run the maven build till your targeted phase.

If you use the option keep java, do not setup your tCompile java directory to your maven project java directory: project/src/main/java or maven compiler will also compile your java files when reaching the phase compile.

If you run the build till the package or install phase, you will have your classes packaged into a jar file named with the name of your project, i.e BUILD_T24.jar, if the classes folder is matching your project/target/classes directory.

Mixed compilation

What we call the "mixed compilation" is a compilation where tCompile is going to generate the java files only and maven compiler will compile them.

This compilation is faster than the classic compilation as there will be only one javac compilation invoked for all java files.

For medium size projects maven compiler should be able to compile all java files in a row, but for high size project, like full T24 compilation, we will need to use specific compiler to do incremental compilation.

The following “translate.only” profile does this mixed compilation.

```
<profile>
      <id>translate.only</id>
            <build>
            <plugins>
            <plugin>
                  <groupId>com.temenos.tafj</groupId>
                  <artifactId>tafj-maven-plugin</artifactId>
                  <executions>
                  <execution>
                        <phase>generate-sources</phase>
                        <goals>
                        <goal>compile</goal>
                        </goals>
                        <configuration>
                              <tafjHome>${tafjHome}</tafjHome>
                              <tafjProperties>tafj</tafjProperties>
                              <basicDir>${basedir}/src/basic/BP</basicDir>
                              <insertDir>$
                        {basedir}/src/basic/BP</insertDir>
                              <!-- Don't change javaDir, has to be
                              src/main/java to have maven compiler finding
                              source code -->
                              <javaDir>${basedir}/src/main/java</javaDir>
                              <classesDir>$
                        {basedir}/target/classes</classesDir>
                              <javaPackage>com.temenos.t24</javaPackage>
                              <properties>
      <temn.tafj.compiler.generate.class>false</temn.tafj.compiler.generate
      .class>
      <temn.tafj.compiler.generate.java>true</temn.tafj.compiler.generate.j
ava>
      <temn.tafj.directory.precompile>$
{lib.dir}</temn.tafj.directory.precompile>
                              </properties>
                        </configuration>
                  </execution>
                  </executions>
            </plugin>
            <!-- GENERATE COMPONENT CLASSPATH -->
            <plugin>
            <groupId>com.googlecode.addjars-maven-plugin</groupId>
            <artifactId>addjars-maven-plugin</artifactId>
            <version>1.0.5</version>
            <executions>
            <execution>
                  <phase>process-resources</phase>
                  <goals>
                  <goal>add-jars</goal>
                  </goals>
                  <configuration>
                  <resources>
                  <resource>
```

```
                  <directory>${lib.dir}</directory>
                  <includes>
                        <include>**/*.jar</include>
                  </includes>
                  </resource>
                  </resources>
                  </configuration>
            </execution>
            </executions>
            </plugin>
            <!-- compile java -->
            <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                  <compilerId>eclipse</compilerId>
                  <source>1.6</source>
                  <target>1.6</target>
                  <failOnError>true</failOnError>
            </configuration>
            <dependencies>
            <dependency>
            <groupId>org.codehaus.plexus</groupId>
            <artifactId>plexus-compiler-eclipse</artifactId>
                  <version>2.2</version>
                  <scope>runtime</scope>
            </dependency>
            </dependencies>
      </plugin>
      </plugins>
      </build>
</profile>
```

We have same kind of configuration as in the classic compilation except the following points:

The java dir much match the maven project source directory to have maven compiler compiling

<!-- Don't change javaDir, has to be src/main/java to have maven compiler finding source code -->

<javaDir>${basedir}/src/main/java</javaDir>

You have to set property «translate only » and «keep java » to true and reference the jars you may have dependencies on with the precompile property.

```
      <temn.tafj.compiler.generate.class>false</temn.tafj.compiler.generate
      .class>
      <temn.tafj.compiler.generate.java>true</temn.tafj.compiler.generate.j
ava>

      <temn.tafj.directory.precompile>$
{lib.dir}</temn.tafj.directory.precompile>
```

To have maven compiler resolving the dependencies, we use the plugin

```
<groupId>com.googlecode.addjars-maven-plugin</groupId>
<artifactId>addjars-maven-plugin</artifactId>
<version>1.0.5</version>
```

It will add to the maven classpath all jars contained in

```
<directory>${lib.dir}</directory>
<includes>
      <include>**/*.jar</include>
</includes>
```

As mentioned above we also need to tune the maven compiler, to be able to compile high size project, to use “eclipse plexus compiler”.

```
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
….
      <compilerId>eclipse</compilerId>

…

<dependency>
<groupId>org.codehaus.plexus</groupId>
<artifactId>plexus-compiler-eclipse</artifactId>
      <version>2.2</version>
      <scope>runtime</scope>
</dependency>
```

You won’t have to tune maven compiler for medium size project and you could remove the maven compiler plugin section.

Simply launch command

mvn clean compile -P translate.only

tafj-maven-plugin will generate the java files and maven compiler compile them.

If you run the build till the package or install phase, you will have your classes packaged into a jar file named with the name of your project, i.e BUILD_T24.jar, if the classes folder is matching your project/target/classes directory.

TAFJ-Runner

Introduction

tRun is the main entry point for running a program. This is a script in the /bin directory. It is used to execute the Compiled Basic Files

tRun

Syntax

The tRun syntax is the following:

```
tRun [-cf <configuration>[.properties]] <BASIC program>
                        [<parameters>]
```

TAFJ Classpath

TAFJ is managing the classpath base on the properties file of the project. There are no need to setup the CLASSPATH variable environment.

The order of the TAFJ Classpath is :

1. All classes specify with the key: **<temn.tafj.directory.classes>**
2. All jars and folder specify with the key: **<temn.tafj.runtime.classpath>**
3. All jars and folder and subfolder in ***<TAFJ_HOME>/ext***
4. All jars and folder specify with the key: **<temn.tafj.runtime.extended.classpath>**
5. All T24 jars and folder specify with the key **<temn.tafj.directory.precompile>**

You can specify multiple paths separate with the separator path. “:” or “;”. Except for the key **<temn.tafj.directory.classes>** only 1 folder can be specify.

I.e.

```
temn.tafj.directory.precompile=/T24/Modules/GB0003:/T24/lib
```

Each entry has a clear state and the defined order cannot be change.

<temn.tafj.directory.classes>

Specify where the classes' files will be generated by the compiler.

<temn.tafj.runtime.classpath>

Specify where all internal jars and classpath before ext folder (3rd party jars and jdbc drivers) will be loaded.

<TAFJ_HOME>/ext

Specify where all 3rd party jars, jdbc drivers and subfolder classpath will be loaded.

<temn.tafj.runtime.extended.classpath>

Specify where all internal jars and classpath after ext folder (3rd party jars and jdbc drivers) will be loaded.

<temn.tafj.directory.precompile>

Specify where all T24, components and modules jars and classpath will be loaded.

TAFJ Runtime Directories

TAFJ support UD tables. The root of the current directory where all UD tables is specify with the key : **"temn.tafj.runtime.directory.current".**

In the properties file the default is :

```
# Specify what will be considered as the "current" directory (eg in an OPEN "." ...)
#
temn.tafj.runtime.directory.current         = <tafj.home>/UD
```

The default file encoding is UTF-8. You can spefify the file encoding of the UD Table with the key : **"temn.tafj.runtime.ud.encoding".**

In the properties file the default is:

```
# By default all UD Tables is no UTF-8 encoding.
# you have to specify the codepage of UD tables
# Latin character ISO-8859-1, cp1252 or UTF-8
temn.tafj.runtime.ud.encoding                       = UTF-8
```

TAFJ treat differently the directory &COMO&. You can specify where is the &COMO& folder outside the current directory of the UD Tables with the key : **“temn.tafj.runtime.directory.como”.**

```
# Specify what is the COMO directory.
#
temn.tafj.runtime.directory.como                    =
<temn.tafj.runtime.directory.current>/&COMO&
```

The default file encoding (empty key) is the platform specific. You can specify the file encoding of the UD Table with the key : **“temn.tafj.runtime.ud.encoding”.**

```
# set the file encoding of the output file COMO
# ex for zOS : temn.tafj.runtime.como.encoding= IBM-1047
# default is ISO-8859-1
#
#temn.tafj.runtime.como.encoding                    = IBM-1047
```

TAFJ treat differently the directory &HOLD&. You can specify where is the &HOLD& folder outside the current directory of the UD Tables with the key : “temn.tafj.runtime.directory.hold**”.**

```
#Specify where is the &HOLD& directory when SETPTR is used with the HOLD option
#
temn.tafj.printer.directory.hold                    =
<temn.tafj.runtime.directory.current>/&HOLD&
```

TAFJ Timezone and Local

`TimeZone` represents a time zone offset, and also figures out daylight savings. You can set the Timezone with the key: “temn.tafj.runtime.timezone”.

In the properties file the default is:

```
# set the timezone of the system
#
#ie : temn.tafj.runtime.timezone = Europe/London
temn.tafj.runtime.timezone                  =
```

A `Locale` object represents a specific geographical, political, or cultural region. An operation that requires a `Locale` to perform its task is called *locale-sensitive* and uses the `Locale` to tailor information for the user. You can set the Local with the key: **“temn.tafj.runtime.local**”.

In the properties file the default is:

```
# set the locale language and country
#
```

```
temn.tafj.runtime.local                    = en_US
```

TAFJ Thread or Process

By convention a JBC command : EXECUTE PHANTOM will forks a new OS Process with a new instance of a JVM (Java Virtual Machine). For performance raison you can define with the key **"temn.tafj.runtime.phantom.as.process"** to execute the new process as a thread in the current JVM.

In the properties file the default is:

```
# Specify whether a "EXECUTE PHANTOM ..." will be a new process or
# a new Thread. In multiple CPU machines, a new process (true) could
# be more efficient.
#
temn.tafj.runtime.phantom.as.process        = true
```

When the new process forks a new JVM, you have to specify the environment of the JVM with options. Use the key: "temn.tafj.runtime.new.process.params"
The option can be any java options:

```
Usage: java [-options] class [args...]
where options include:
    -d32                          use a 32-bit data model if available
    -d64                          use a 64-bit data model if available
    -server                       to select the "server" VM
    -hotspot                      is a synonym for the "server" VM  [deprecated]
                                  The default VM is server.
    -D<name>=<value>     set a system property
    -verbose:            [class|gc|jni] enable verbose output
    -ea[:<packagename>...|:<classname>]
    -enableassertions[:<packagename>...|:<classname>]
                  enable assertions with specified granularity
    -da[:<packagename>...|:<classname>]
    -disableassertions[:<packagename>...|:<classname>]
                  disable assertions with specified granularity
    -esa | -enablesystemassertions
                  enable system assertions
    -dsa | -disablesystemassertions
                  disable system assertions
    -agentlib:<libname>[=<options>]
                  load native agent library <libname>, e.g. -agentlib:hprof
                  see also, -agentlib:jdwp=help and -agentlib:hprof=help
    -agentpath:<pathname>[=<options>]
                  load native agent library by full pathname
    -javaagent:<jarpath>[=<options>]
                  load Java programming language agent, see java.lang.instrument
    -splash:<imagepath>
                  show splash screen with specified image

See http://www.oracle.com/technetwork/java/javase/documentation/index.html for more details.
```

In the properties file the default is:

```
#If the phantoms are executed as process, what JVM options to
#use to launch them.
#
temn.tafj.runtime.new.process.params        = -Xmx1024M -XX:MaxPermSize=256m
```

TAFJ Precision and Rounding

By properties, you can specify the default precision for the runtime with the key: **“temn.tafj.runtime.default.precision”.**

In the properties file the default is:

```
# Default precision when not specified.
#
temn.tafj.runtime.default.precision         = 4
```

By properties, you can specify the rounding of a operation with the key: **“temn.tafj.runtime.rounding.mode”** and the rounding of the result after operation with the key : **“temn.tafj.runtime.rounding.mode.result”.**

In the properties file the default is:

```
# Rounding Mode for operands : HALF_UP(Default) (2.25 -> 2.3 and 2.24 -> 2.2)
# Possible values : HALF_DOWN, HALF_UP, HALF_EVEN, CEILING, FLOOR, UP
#
temn.tafj.runtime.rounding.mode                     = HALF_UP

# Rounding Mode result : HALF_UP(Default) (2.25 -> 2.3 and 2.24 -> 2.2)
# Possible values : HALF_DOWN, HALF_UP, HALF_EVEN, CEILING, FLOOR, UP
#
temn.tafj.runtime.rounding.mode.result              = DOWN
```

TAFJ Runtime Mode

If TAFJ is install on a Mainframe (zOS), you need to specify to true the key: **“temn.tafj.runtime.zos.mode”.** It will avoid the INPUT statement with timer. (not allow on zOS). Set the correct encoding for logger in TAFJTrace.properties and COMO encoding.

In the properties file the default is:

```
# set to true if the runner is on a zOS LPar machine
# and it have to work in legacy mode
#
temn.tafj.runtime.zos.mode          = false
```

Here is some other specific mode.

```
# Define if, in case of exception, we are stopping the execution or
# just throwing the exception (in case of TAFJ is used like an API)
#
temn.tafj.runtime.exception.mode                    = false

# When set to true a terminating session will be cleaned up.
# All locks released and the database connection closed.
#
temn.tafj.runtime.clean.on.shutdown         = true

# By default data files loaded require manual authorization.
# When set to true data are uploaded in $INAU table.
# When set to false data are uploaded directly in the LIVE table
temn.tafj.runtime.authorize.record          = true
```

TAFJ Performance

For Performance raison, and if you are not in development environment you have to active some cache and stop TAFJ to have the capability to debug code.

In the properties file the default is:

```
# If set to false, any 'DEBUG' statement will be ignored.
#
temn.tafj.runtime.enable.debug              = true

# Performance :cache OPF
#
temn.tafj.runtime.enable.cached.opf         = false

# Performance : internal java TAFJ System_getCache
#
temn.tafj.runtime.use.cache.get                     = false
```

TEC and logger API

TAFJ has the capability to redirect API message to the logger API (T24) to the JMS Topic Queue with the key : **“temn.tafj.runtime.enable.logger.api.jms”**

In the properties file the default is:

```
# Enable the Logger api for TEC Items recording using JMS (used by T24Monitor)
#
temn.tafj.runtime.enable.logger.api.jms             = false
```

TAFJ has the capability to disable any T24 Logger API or TEC. With the key: **“temn.tafj.runtime.enable.logger.api“** and the key: **“temn.tafj.runtime.disable.tec“**

In the properties file the default is:

```
# Enable the Logger api for TEC Items (used by T24Monitor)
#
temn.tafj.runtime.enable.logger.api  = true

# Enable TEC Items recording
#
temn.tafj.runtime.disable.tec               = false
```

TAFJ JIMI (Independent Metrics Integration)

Check the below properties for JIMI.

In the properties file the default is:

```
#*********************************************************************
#
# JIMI
#
#*********************************************************************

# Specify if jimi is on, off or can be switched
# 1 : On, can be changed programatically
# 0 : Off, can be changed programatically
#-1 : Off CANNOT be changed programatically.
#
temn.tafj.runtime.jimi.on                           = 0

# Specify what directory will contains the jimi reports
#
temn.tafj.runtime.directory.jimi                    = <temn.tafj.runtime.directory.current>

# Specify whether we want to append on existing trace files or create
# a new file every times we start jimi.
#
temn.tafj.runtime.jimi.append                       = false

# The first index when using the FILEIO option
#
temn.tafj.runtime.jimi.image.index                  = 1
```

TAFJ Monitor

To use TAFJ Monitor, TAFJSessionMonitor have to be up and running on the network. TAFJ with the following properties will send all the information to the TAFJSessionMonitor. Read the section TAFJSessionMonitor to know how setup and start the monitor.

```
#*********************************************************************
#
#  TAFJ Monitor
#
#*********************************************************************
# Enable the TAFJMonitorSession
#
temn.tafj.runtime.session.monitor.enable        =          true

# Host name or IP of where is the TAFJMonitorSession
#
temn.tafj.runtime.session.monitor.host              =              localhost

# TCP port of the TAFJMonitorSession
#
temn.tafj.runtime.session.monitor.port              =              8377
```

TAFJ Printer

TAFJ use the JPS (Java Print Service.

On Windows, JPS use the Printer Manager. You can connect any printers installed on Windows. The Printer has to be online.

On UNIX/LINUX, JPS use CUPS (Common Unix Printing System). CUPS is not install by default. You have to install it. Please refer to your OS documentation for the installation.

In the properties file, category printer, the setup of TAFJ automatically adds your printers online with default driver name PRN #. You can affect the channel to any drivers' name.

You can affect 256 channels to different printer with the key: **"temn.tafj.channel.name.#". #** is the number of the channel.

The name you affect to the key have to match the name of the driver.

i.e

```
temn.tafj.channel.name.0  = PRN2
```

The channel 0 is link to the driver PRN2

To install a driver, you need to specify:

1) A name to be link with the channel
2) A device base on the printer manager of windows or CUPS printer name.
3) A class driver (by default : com.temenos.tafj.jlp.drivers.jPrinterDriver)

To find the devices online on your OS, use the command tFindDevice.

```
_____________________________________________________________________

Device(s) on your system
_____________________________________________________________________

Default printer : >HP Officejet Pro 8600 (Network)<
Printer 0       : >Send To OneNote 2010<
Printer 1       : >Microsoft XPS Document Writer<
Printer 2       : >HP0CE4D2 (HP Officejet Pro 8600)<
Printer 3       : >HP Officejet Pro 8600 (Network)<
Printer 4       : >Fax - HP Officejet Pro 8600 (Network)<
Printer 5       : >Fax<

_____________________________________________________________________
```

Chose the device you want and add to the properties file:

```
temn.tafj.driver.name.#              = <The name of a key to link with the channel>
temn.tafj.driver.device.#     = <The name of device>
temn.tafj.driver.class.#      = com.temenos.tafj.jlp.drivers.jPrinterDriver
```

is the next available number of the device.

i.e.

```
temn.tafj.channel.name.0      = PRN0
temn.tafj.channel.name.1      = PR23
temn.tafj.channel.name.2              = PRN1

temn.tafj.driver.name.0               = PRN0
temn.tafj.driver.device.0     = HP Officejet Pro 8600 (Network)
temn.tafj.driver.class.0      = com.temenos.tafj.jlp.drivers.jPrinterDriver

temn.tafj.driver.name.1               = PRN1
temn.tafj.driver.device.1     = Send To OneNote 2010
temn.tafj.driver.class.1      = com.temenos.tafj.jlp.drivers.jPrinterDriver

temn.tafj.driver.name.2               = PRN2
temn.tafj.driver.device.2     = Microsoft XPS Document Writer
temn.tafj.driver.class.2      = com.temenos.tafj.jlp.drivers.jPrinterDriver
```

TAFJ DataBase

TAFJ use JDBC to connect and use a database. Short for ***J**ava **D**atab**b**ase **C**onnectivity,* a Java API that enables Java programs to execute SQL statements. This allows Java programs to interact with any SQL-compliant database. Since nearly all relational database management systems (DBMSs) support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs. JDBC is similar to ODBC, but is designed specifically for Java programs, whereas ODBC is language-independent.

You need to setup 4 keys to connect to the database:

temn.tafj.jdbc.url

URL that describes database connection

temn.tafj.jdbc.driver

Class that describes the specific Driver for a database

temn.tafj.jdbc.username

The username to connect

temn.tafj.jdbc.password

The password

i.e.

```
#*********************************************************************
#
# Database setup
#
#*********************************************************************

# URL that describes database connection,
# ex.  oracle:           jdbc:oracle:thin:@<host_or_ip>:1521:<db_name>
# ex.  db2:              jdbc:db2://<host_or_ip>:50000/<db_name>
# ex.  ms-sql:           jdbc:sqlserver://<host_or_ip>:1433;databaseName=<db_name>;integratedSecurity=true
# ex.  H2:               jdbc:h2:tcp://<host_or_ip>/<db_name>
# ex.  H2(Embedded):     jdbc:h2:<path_to_my_db>
#
temn.tafj.jdbc.url    = jdbc:db2://158.35.69.124:50000/t24mb

# Class that describes the specific Driver for a database,
# ex. oracle:                 oracle.jdbc.driver.OracleDriver
# ex. db2:             com.ibm.db2.jcc.DB2Driver
# ex. ms-sql:                 com.microsoft.sqlserver.jdbc.SQLServerDriver
# ex. H2:                org.h2.Driver
#
temn.tafj.jdbc.driver = com.ibm.db2.jcc.DB2Driver

temn.tafj.jdbc.username       = tafjdb
temn.tafj.jdbc.password       = secret
```

!! WARNING !!

You need the specific database drivers you are targeting to go in ***<TAFJ_HOME>*/ext**.
Please find these drivers in ***<TAFJ_HOME>*/dbdrivers** and copy them to ***<TAFJ_HOME>*/ext**

i.e for oracle 11g database

```
<TAFJ_HOME>
    ├──dbdrivers
    │    ├──db2_v10.1
    │    ├──db2_v10_zOS
    │    ├──h2_1.3.161
    │    ├──h2_1.3.172
    │    ├──oracle-11g
    │    ├──oracle-12c
    │    └──sqljdbc_1.2
    ├──ext
    │    └──oracle-11g
```

TAFJ LockManager

To configure the client system to use the Distributed Lock Service, each client user must be configured with the **TAFJ Locking mechanism**.

To be properly effective ALL users of the same database must be configured with exactly the same
Distributed Lock Service parameters, otherwise locks will NOT be correctly respected and data inconsistencies may occur.

You need to setup the following keys :

temn.tafj.locking.mode

TAFJ Distributed Lock mechanism mode.

temn.tafj.locking.hostname

Hostname or the IP address of the system where the Distributed Lock Service is executing.

temn.tafj.locking.port

The socket port number on which TAFJ Distributed Lock service is listening.

temn.tafj.locking.callstack

Bring the current call stack over to the LockManager?

temn.tafj.locking.name

override the URL as unique key for the DBInstance

TAFJ Locking mechanism .

1. MEM :
   The LockManager is part of the same JVM than the process running.
   - High Performance.
   - ⚠ **The runtime have to be in thread mode.**
   - ⚠ **The locks cannot be share between multiple JVM**

2. JDBC : use the same URL than the database
   The LockManager is a table in the database. The name of the table use for the lock is LOCK_RECORDS
   - Best situation for development purpose.
   - ⚠ **Poor performance.**

3. PROC / PROCHASH
   The LockManager is an external process on a server in the network.
   The key "**temn.tafj.locking.hostname"** and **"temn.tafj.locking.port"** have to be set.
   The PROCHASH is the same mechanism as PROC but the lock is a hash. It cannot be monitor.
   - Good Performance.
   - Solution for DB2 database
   - ⚠ **tLockManager have to be started**

4. ORCL / MSQL
   The LockManager use the lock engine of the database via an API.
   - Good Performance.
   - ⚠ Solution only for Oracle and MS-SQL Server
   - ⚠ **This is not the native locks of the database.**

TAFJ Tools

tDiag

Introduction

tDiag is part of the TAFJ tools. It allows to easily get information about your environment like the java version, the TAFJ version,… The tDiag utility is used to display information about your TAFJ installation and projects setup. It is useful in pin-pointing problem areas of the setup.

Syntax

The tDiag syntax is the following:

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

Example

To get information about all projects you have configured

```
tDiag
```

The resulting output looks like this:

```
------------------------------------------------------------------

Home                      : 'D:\Temenos\T24Enterprise\TAFJ'
Conf directory            : 'D:\Temenos\T24Enterprise\TAFJ/conf'
Log directory             : 'D:\Temenos\T24Enterprise\TAFJ/log'
Version                   : DEV_201404
------------------------------------------------------------------

java.home                 : D:\Temenos\T24Enterprise\3rdParty\java\jdk1.7.0_51-64\jre
java.vendor               : Oracle Corporation
java.version              : 1.7.0_51
os.arch                   : amd64
os.name                   : Windows 7
HostName                  : wkshome01
IP Address                : 10.244.1.137
Runtime                   : Double Byte SysSeparator
------------------------------------------------------------------

Default Project           : 'tafj'

- Project : 'tafj'
        Basic source             : D:\Temenos\T24Enterprise\TAFJ\samples\basic
        Java src dir             : D:\Temenos\T24Enterprise\TAFJ\data\tafj\java
        Java classes dir         : D:\Temenos\T24Enterprise\TAFJ\data\tafj\classes
        Update dir               : D:\Temenos\T24Enterprise\TAFJ/updates
        ---------------------------------------------------------------
        Java default package     : com.temenos.t24
        ---------------------------------------------------------------
        DataBase URL             :
        DataBase user            :
        ---------------------------------------------------------------
        Locking mode             : JDBC
        Locking name             :
        ---------------------------------------------------------------
        Current dir              : D:\Temenos\T24Enterprise\TAFJ/UD
        Como dir                 : D:\Temenos\T24Enterprise\TAFJ/UD/&COMO&
        Hold dir                 : D:\Temenos\T24Enterprise\TAFJ/UD/&HOLD&
        UD encoding              : UTF-8
        ---------------------------------------------------------------
        Timezone                 : Europe/London
        Local                    : en_US
        Debug enabled            : true
        ---------------------------------------------------------------
        JMS logger enabled       : false
        Logger API enabled       : true
        TEC disabled             : false
        ---------------------------------------------------------------
        Session monitor enabled : false
        Session monitor host     : localhost
        Session monitor port     : 8377
```

tShow

Introduction

tShow is part of the TAFJ tools. It allows finding easily where your BASIC source has been compiled, when and on what machine it has been compiled. It can search for the resource on one single project and multiple ones.It can also produce a report of all the duplicate classes when run with –d argument.

Syntax

The tShow syntax is the following:

```
tShow [-cf <configuration>[.properties]] <SUBROUTINE.NAME>

tShow [-cf <configuration>[.properties]] -d
```

Example

To view where CACHE.READ is, just type

```
tShow CACHE.READ
```

It will, for all projects in your distribution, try to load the java class for CACHE.READ (CACHE_READ_cl.class), and get information like:

- Where is the original BASIC source
- When it has been compiled
- On what machine is has been compiled
- What grammar applied to this file
- Whether it is having Basic Replacement

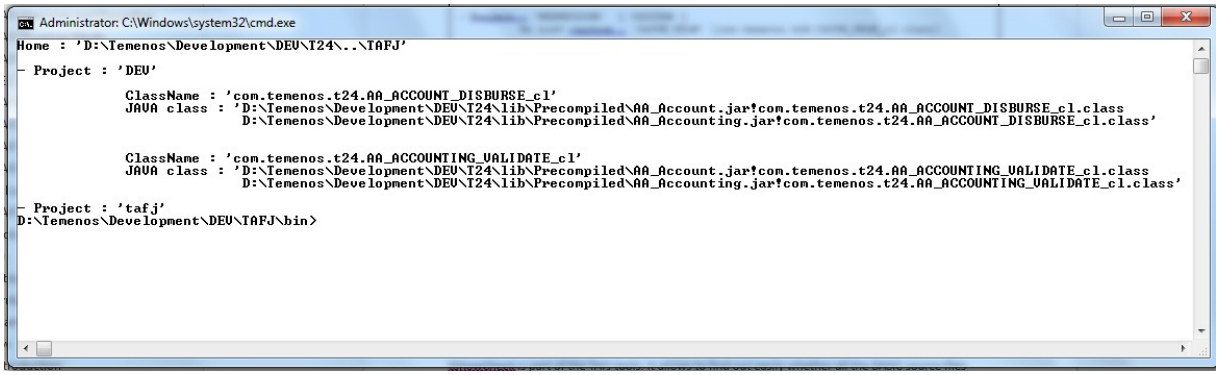
Here is the result:

```
Home : 'D:\Temenos\T24Enterprise\TAFJ'

- Project : 'TAFJ201305'  [ FOUND  ]
             BASIC source : 'C:\Product\Version\201305\T24_BP\CACHE.READ'
            BASIC package : ''
          BASIC Import(s) : ''
               JAVA class : 'file:/
'C:\Product\Version\201305/lib/201305.jar!/com/temenos/t24/CACHE_READ_cl.class'
             Compiled the : 12 Jul 2013 07:48:27
                       on : GVAL1201010
       Compiled with TAFJ : < R13GA
                Timestamp : 1342072107042
                  Grammar : 1
Include Basic Replacement : false

- Project : 'REGRESSION'  [ MISSING ]
          No such routine : 'CACHE.READ' (com.temenos.t24.CACHE_READ_cl.class)
```

To view the Duplicate classes Report,

```
tShow -d
```

Here is the result:



tShowCheck

Introduction

tShowCheck is part of the TAFJ tools. It allows to find out easily whether all the BASIC source files have been compiled successfully. It reports the error and warning messages to a flat file if a class is found missing in the precompiled or in the classes folder.

It takes three command line parameters,

- -s <PATH_OF_SOURCE_FOLDER>

- -p <PATH_OF_PRECOMPILED> or <PATH_OF_CLASSES_FOLDER>
- -r <PATH_OF_REPORT_FOLDER>

Syntax

The tShowCheck syntax is the following:

```
tShowCheck [-cf <configuration>[.properties]] –s [Path_of_source_folder] –p
[<path_of_precompiled> or <path_of_classes_folder>] –r [path_of_report_folder]
```

Example

```
tShowCheck  -s D:\T24_BP –p D:\T24_Precompiled\R10GA.jar –r d:\Report
```

It will generate new report file with name of tShowReport.txt

```
No such routine : 'SYSTEM.SANITY.CHECK'
(com.temenos.t24.SYSTEM_SANITY_CHECK_cl.class) : [ MISSING ]
 Class found but failed to get informations for 'T.TRACE'
(com.temenos.t24.T_TRACE_cl.class) : [ WARNING ]
 No such routine : 'TEMPLATE' (com.temenos.t24.TEMPLATE_cl.class) : [ MISSING ]
 No such routine : 'TSDK.CALLJ.PROG.JBASE'
(com.temenos.t24.TSDK_CALLJ_PROG_JBASE_cl.class) : [ MISSING ]
 No such routine : 'TV.GET.AA.CHILD.TRANSACTION'
(com.temenos.t24.TV_GET_AA_CHILD_TRANSACTION_cl.class) : [ MISSING ]
 No such routine : 'TWS.AA.GET.PROPERTY.TEMPLATES'
(com.temenos.t24.TWS_AA_GET_PROPERTY_TEMPLATES_cl.class) : [ MISSING ]
 No such routine : 'Tws.isTws' (com.temenos.t24.Tws_isTws_6_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA' (com.temenos.t24.TWS_SCHEMA_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA.APPLICATION'
(com.temenos.t24.TWS_SCHEMA_APPLICATION_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA.ENQUIRY'
(com.temenos.t24.TWS_SCHEMA_ENQUIRY_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA.GENERATE'
(com.temenos.t24.TWS_SCHEMA_GENERATE_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA.PROCESS'
(com.temenos.t24.TWS_SCHEMA_PROCESS_cl.class) : [ MISSING ]
 No such routine : 'TWS.SCHEMA.VERSION'
(com.temenos.t24.TWS_SCHEMA_VERSION_cl.class) : [ MISSING ]
 No such routine : 'UPDATE.MVS.SVS.IN.FVS'
(com.temenos.t24.UPDATE_MVS_SVS_IN_FVS_cl.class) : [ MISSING ]
 No such routine : 'User.getCompanyList'
(com.temenos.t24.User_getCompanyList_15_cl.class) : [ MISSING ]
 No such routine : 'User.validateCompanySwitch'
(com.temenos.t24.User_validateCompanySwitch_22_cl.class) : [ MISSING ]
 No such routine : 'Util' (com.temenos.t24.Util_3_cl.class) : [ MISSING ]
 No such routine : 'V.AA.ARR.DEFAULT.CUSTOMER'
(com.temenos.t24.V_AA_ARR_DEFAULT_CUSTOMER_cl.class) : [ MISSING ]
 No such routine : 'V.AM.COMPARE.VER.AUTHRTN'
(com.temenos.t24.V_AM_COMPARE_VER_AUTHRTN_cl.class) : [ MISSING ]
 No such routine : 'V.MB.CHILD.ACCOUNT.RULES'
(com.temenos.t24.V_MB_CHILD_ACCOUNT_RULES_cl.class) : [ MISSING ]
 No such routine : 'V.MB.COMI.PROCESS' (com.temenos.t24.V_MB_COMI_PROCESS_cl.class)
: [ MISSING ]
 No such routine : 'V.MB.CUST.PROCESS' (com.temenos.t24.V_MB_CUST_PROCESS_cl.class)
: [ MISSING ]
 No such routine : 'V.MB.STO.CREATE.BAL'
(com.temenos.t24.V_MB_STO_CREATE_BAL_cl.class) : [ MISSING ]
 No such routine : 'V.MB.STO.CREATE.FIX'
(com.temenos.t24.V_MB_STO_CREATE_FIX_cl.class) : [ MISSING ]
************************************************************************************
*
Total Number of Routines Requested                    : 17929
Total Number of Routines Found                        : 17419
Total Number of Routines Failed                       : 505
Total Number of Routines with BasicReplacements       : 5
Total Time taken                                     : 0 Hrs 2 Mins 54 Secs
```

tCrypt

Introduction

tCrypt is part of the TAFJ tools. It allows encrypting the password specified in temn.tafj.jdbc.password. This will encrypt the password with DES3 Algorithm. The key used to encrypt is auto generated and stored in a file called .key in the conf directory of your TAFJ installation.

Syntax

The tCrypt syntax is the following:

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

Example

To get encrypt the password for the database mentioned in configuration file

```
tCrypt -cf tafj
```

The resulting output looks like this:

```
C:\TAFJ\bin>tCrypt
Configuration: 'C:\TAFJ\bin\..\conf\tafj.properties'
Configuration file changed with encrypted password.
C:\TAFJ\bin>
```

tafj.properties will then have the password looking like that :

Before tCrypt, password will look like this:

```
temn.tafj.jdbc.username             = t24
temn.tafj.jdbc.password             = secret
```

After tCrypt, password changes will look like this:

```
temn.tafj.jdbc.username             = t24
temn.tafj.jdbc.password             = ::P6PeGkfV134=
```

tFindDevice

Introduction

tFindDevice is part of the TAFJ tools. It allows to easily getting information about your Default Printer on your System and a list of printers which are available on your system.

Syntax

The **tFindDevice** syntax is the following:

```
tFindDevice
```

Example

To Find the Default printer on your system

```
tFindDevice
```

The resulting output looks like this

```
_____________________________________________________________________

Device(s) on your system
_____________________________________________________________________

Default printer : >HP Officejet Pro 8600 (Network)<
Printer 0       : >Send To OneNote 2010<
Printer 1       : >Microsoft XPS Document Writer<
Printer 2       : >HP0CE4D2 (HP Officejet Pro 8600)<
Printer 3       : >HP Officejet Pro 8600 (Network)<
Printer 4       : >Fax - HP Officejet Pro 8600 (Network)<
Printer 5       : >Fax<

_____________________________________________________________________
```

tCreateBasicReplacement

Introduction

This chapter will cover how you can easily and safely write java code and invoke it directly from BASIC. You can write your own java class, and setup the TAFJ runtime to use it instead of a BASIC subroutine. You also can simply invoke your java class without having a corresponding BASIC subroutine. Of course, the whole environment (like COMMONs) is available in your java class.

Writing your java class

The class you will invoke from basic (via a standard CALL(...)) must extends ***com.temenos.tafj.runtime.extension.BasicReplacement***.

This class must also implement a static method “INSTANCE” and a public method stack.

In order to make your live easier, and to avoid the copy/paste paradigm, there is a tool available in the ***<TAFJ_HOME>/bin*** directory : ***tCreateBasicReplacement (.bat)***

Syntax

The **tCreateBasicReplacement** syntax is the following:

```
tCreateBasicReplacment <full package_name_class>
```

Example

To create a java MailSender

```
tCreateBasicReplacement com.temenos.t24.MailSender
```

The above command will create, in the ext directory, the correct file structure and the class template you specified :

```
ext/
 |_ com/
    |_ temenos/
        |_ t24/
           |_ MailSender.java
```

The class MailSender contains all the necessary information for being invoked from BASIC. You will need to add your custom code in the method invoke(Object... args).

The method invoke

```
public jVar invoke(Object... args) {
    //TODO : Add your code here.
    String sMessage = args[0].toString();
    ((jVar)args[1]).set("Response 3 : " + sMessage);
    return null;
}
```

The argument of this method is an array of objects. These objects are in fact jVar (a variable), and you can cast them to get the correct reference.

Despite the fact that the TAFJ jVar object is not part of any API, here are few useful methods :

- to know how many parameters are passed : args.length
- to convert the var in String : toString()
- to set a value to a var (eg. for returning a value in BASIC) : set(String)
- to get a specific FieldMark, ValueMark, SubvalueMark : get(<fm>, <vm>, <sm>)

There are MANY methods available, but this is not the purpose of this document to list them all.

Compiling the class

You can compile this class as any other java class. The only dependency is on the TAFJCore.jar library.

So for example, you can type, from the ***<TAFJ_HOME>/ext*** directory:

```
javac -cp ../lib/* com.temenos.t24.MailSender.java
```

This will create a MailSender.class next to the java file.

Registering a BASIC replacement

Once the class has been implemented (and compiled!) you will need to let the runtime know about it. To do that, edit the property file of your project (default is tafj.properties) and find the basic replacement section. This section looks like this by default :

```
#*********************************************************************
#
# Basic Replacement
#
#*********************************************************************
#
# These properties are shortcuting the CALL statement to invoke
# Directly a javaClass.
# The Parameters are all jVar and the count must match the
# Basic equivalent.
# The count must not be discountinuous. This means that if
# there is ...1, ...2, ...4 without ...3, ...4 will not be read and thus ignored.
#
temn.tafj.directory.ext             = <tafj.home>/ext

temn.tafj.migration.basic.1     = CHECK.ROUTINE.EXIST
temn.tafj.migration.java.1      = com.temenos.tafj.basic.CheckRoutineExist

temn.tafj.migration.basic.2     = EB.CREATE.VIEW
temn.tafj.migration.java.2      = com.temenos.tafj.basic.EbCreateView

temn.tafj.migration.basic.3     = EB.TRACE.CALL
temn.tafj.migration.java.3      = com.temenos.tafj.basic.EbTraceCall

temn.tafj.migration.basic.4   = EBS.CREATE.FILE
temn.tafj.migration.java.4    = com.temenos.tafj.basic.EbsCreateFile
```

As you can see, there is already few basic replacements defined by default. The only thing you need to do is to add yours at the end of the list like this :

```
#*********************************************************************
#
# Basic Replacement
#
#*********************************************************************
#
# These properties are shortcuting the CALL statement to invoke
# Directly a javaClass.
# The Parameters are all jVar and the count must match the
# Basic equivalent.
# The count must not be discountinuous. This means that if
# there is ...1, ...2, ...4 without ...3, ...4 will not be read and thus ignored.
#
temn.tafj.directory.ext             = <tafj.home>/ext

temn.tafj.migration.basic.1     = CHECK.ROUTINE.EXIST
temn.tafj.migration.java.1      = com.temenos.tafj.basic.CheckRoutineExist

temn.tafj.migration.basic.2     = EB.CREATE.VIEW
temn.tafj.migration.java.2      = com.temenos.tafj.basic.EbCreateView

temn.tafj.migration.basic.3     = EB.TRACE.CALL
temn.tafj.migration.java.3      = com.temenos.tafj.basic.EbTraceCall

temn.tafj.migration.basic.4   = EBS.CREATE.FILE
temn.tafj.migration.java.4    = com.temenos.tafj.basic.EbsCreateFile

temn.tafj.migration.basic.5   = SEND.MAIL
temn.tafj.migration.java.5    = com.temenos.t24.MailSender
```

The next time you will compile SEND.MAIL, instead to compile SEND.MAIL, TAFJ Compiler will create a wrapper to call your Basic Replacement.

When TAFJ will execute a CALL SEND.MAIL(...), it will invoke the method invoke(Object... args) of your class com.temenos.t24.MailSender.

Telnet (VT100)

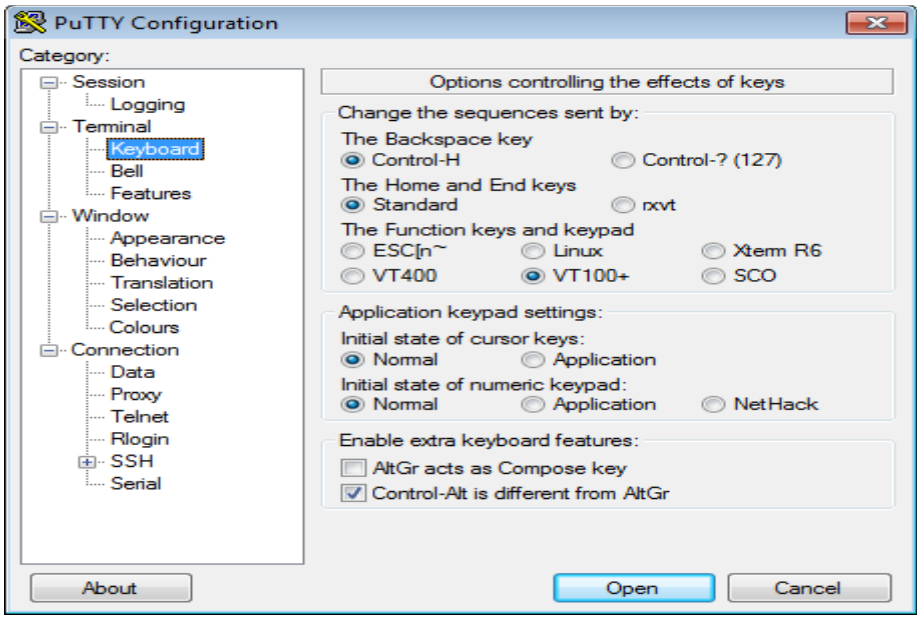
Question: Running tRun EX gives a garbled t24 welcome page with many bad characters on windows 7, why?

Response: By running directly tRun EX in a command window you will not escape telnet characters. You need to run tRun inside a telnet client. The default windows telnet client doesn't render correctly the t24 console page. Using putty, a free telnet client, you get a good rendering.

Install and start the default windows telnet server.

On Windows 7, add your user to TelnetClients group.

Start putty, select telnet protocol, put localhost for the hostname, remove in "Terminal" panel the text inside Answerback ^E field, in "Terminal\keyboard" Ctrl-H for Backspace key and VT100 for Function key.

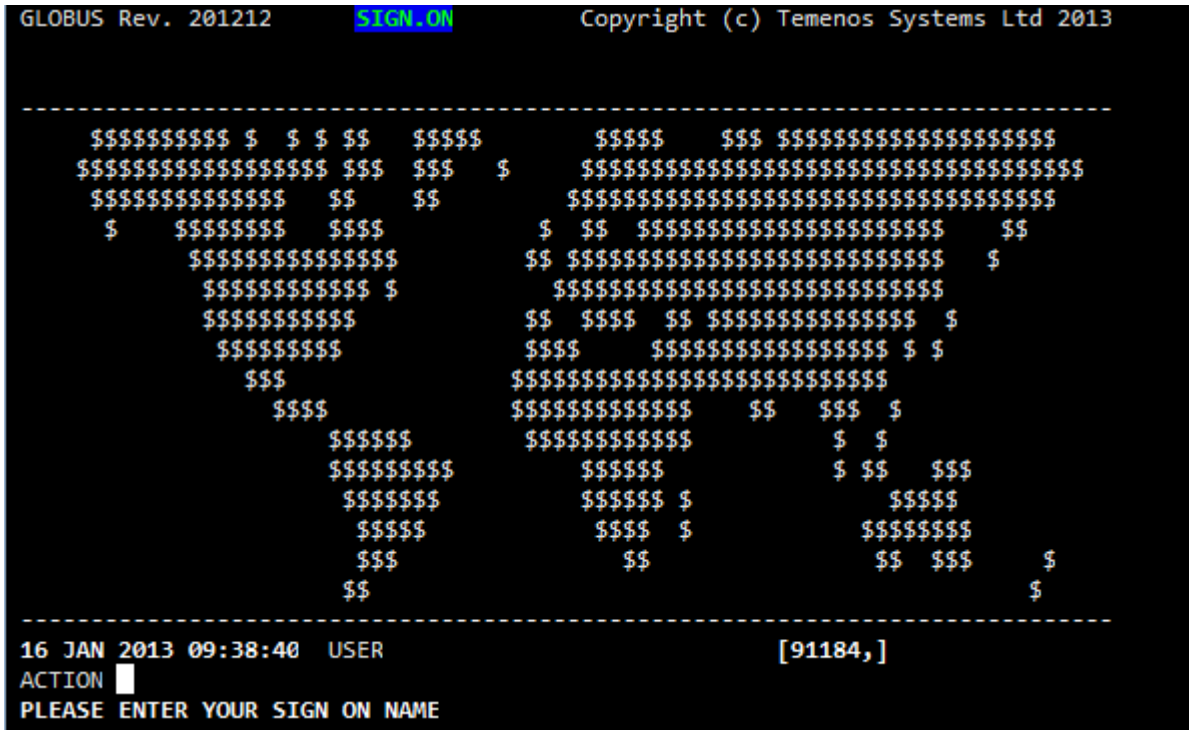


Save your session configuration

Click on open

Go in you TAFJ_HOME\bin

Launch tRun EX.


GLOBUS Rev. 201212
SIGN.ON
Copyright (c) Temenos Systems Ltd 2013
16 JAN 2013 09:38:40 USER
[91184,]
ACTION
PLEASE ENTER YOUR SIGN ON NAME