Oracle Tuxedo Application Runtime for Batch

Reference Guide

11*g* Release 1 (11.1.1.1.0)

March 2010

ORACLE

Oracle Tuxedo Application Runtime for Batch Reference Guide, 11*g* Release 1 (11.1.1.1.0)

Contents:

4. Tuxedo Job Enqueueing Service (TuxJES)

CHAPTER 1

Introduction

The Oracle Tuxedo Application Runtime for Batch normalizes Korn shell script formats by proposing a script model in which the different execution phases are clearly identified, and provides the Tuxedo Job Enqueueing Service (TuxJES), which emulates the major functions of Mainframe JES2.

This guide consists of three main parts:

- The first part describes the equivalencies that exist between JCL cards, general utility commands and sorts on the one hand and the Batch Runtime functions on the other.
- The second part describes how the Korn shell scripts are structured to reproduce a JCL type processing of jobs. The different functions of the Batch Runtime that are used in these scripts are then described in detail.
- The third part describes the servers and utilities for TuxJES.

Z/OS JCL in the Batch Runtime Environment

Introduction to z/OS JCL in the Batch Runtime Environment

This section describes how to find equivalents for z/OS JCL cards in the target environment. Some of these equivalents point to the Batch Runtime functions, other equivalents may rely directly on UNIX or Tuxedo features. In some cases, there may be no equivalent and a work-around solution may be necessary.

It is not the purpose of this document to describe z/OS JCL, for any explanation of JCL statements, please see the z/OS Internet Library.

z/OS JCL Cards in the Batch Runtime Environment

The following tables lists the JCL card parameters and the related command in the Batch Runtime:

JCL Card Equivalence Table

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment
"//*" <comment>		Comments are copied to the Korn shell script.
"//PRO001.LAB1" DD statements		**m_FileOverride**
DD	*	**m_FileAssign and cat function**
	DATA	**Supported**
	DISP	**m_FileAssign -d <DISP option>**
	DLM	**Supported**
	DSNAME	**m_FileAssign**
	DUMMY	**m_FileAssign with /dev/null**
	DYNAM	**m_FileAssign with /dev/null**
	JOBLIB	**m_JobLibSet**
	STEPLIB	**m_StepLibSet**
	SYSIN	**m_FileAssign**
	Printing Parameters	**COPIES: supported**
		DEST: supported
		HOLD: supported
		OUTPUT: supported
		SYSOUT: supported.
EXEC	COND	**m_CondExec**
	PARM	**m_ProgramExec**
	PGM	**m_ProgramExec**
	PROC	**m_ProcInclude**

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment
EXEC (continued)	JCL Symbol definition	**m_SymbolDefault**
IF THEN ELSE END		**m_CondIf** **m_CondElse** **m_CondEndif**
INCLUDE	MEMBER	**m_ShellInclude**
JCLLIB	ORDER	**m_JclLibSet**
JOB	<jobname>	**m_JobBegin**
	CLASS	**m_JobBegin** (with TuxJES interface).
	PRTY	**m_JobBegin** (with TuxJES interface).
	RESTART	**m_JobBegin** (with TuxJES interface).
	TYPRUN	**m_JobBegin** (with TuxJES interface).
OUTPUT	CLASS	**Supported**.
	COPIES	**Supported**.
	DEFAULT	**Supported**.
	DEST	**Supported**.
	FORMS	**Supported**.
	PRTY	**Supported**.
PROC		**m_ProcInclude**
in-stream PROC		**m_ProcBegin**
in-stream PEND		**m_ProcEnd**
SET		**m_SymbolSet**

General Utility Commands Equivalence Table

Table 2-2 General Utility Commands Equivalences

Utility	Command	Equivalent in Target Environment
IDCAMS	ALTER NEWNAME	m_FileRename
	BLDINDEX	m_FileBuild
	DEFINE ALTERNATEINDEX	m_FileBuild
	DEFINE CLUSTER	m_FileBuild
	DEFINE GENERATIONDATAGROUP	m_GenDefine
	DELETE ALTERNATEINDEX	m_Filebuild
	DELETE CLUSTER	m_FileDelete
	DELETE GDG	m_FileDelete
	REPRO	m_FileLoad (Partial)
IEBCOPY	COPY	m_DirCopy **Note:** Only the entire copy from one PDS in input to one PDS in output is available.
IEBGENER		m_FileLoad, m_FileSort (Partial) **Note:** The GENERATE WITH MEMBER command is not supported.
IEFBR14		m_ProgramExec IEFBR14

Sort Utilities

Sort Utilities Equivalence Tables

SORT, SORTD, DFSORT

Table 2-3 Sort Utilities Equivalences

Sort Statement	Parameter	Equivalent in Target Environment
SORT		m_FileSort
MERGE		m_FileSort
OPTION		
	COPY	Supported.
	SKIPREC	Supported.
	STOPAFT	Supported.
	INCLUDE	Supported.
	OMIT	Supported.
	OUTFILE	Supported.
	OUTREC	Supported.
	END	Supported.
	RECORD	Supported.
	SUM	Supported.

ICETOOL

Table 2-4 ICETOOL Equivalences

Sort Statement	Equivalent in Target Environment
SORT	m_FileSort
COPY	m_FileSort

DB2 utilities

Table 2-5 DB2 Utilities

Command	Support
DSNTEP2/4	m_ExecSQL
DSNTIAD	m_ExecSQL

CHAPTER 3

Oracle Tuxedo Application Runtime for Batch Functions

Introduction to the Batch Runtime Commands

This chapter describes:

- The format and rules for using the Batch Runtime Korn shell scripts to run jobs in Emulating z/OS JCL Logic and Architecture.
- The use of the Batch Runtime spawner (EJR) to launch jobs in EJR syntax.
- The log files and return codes used by the scripts and spawner in Log file management and Return code management.
- The use of the Batch Runtime Cobol runtime to trap errors and manage database interaction in Cobol runtime.
- A complete description of the Batch Runtime functions in Oracle Tuxedo Application Runtime for Batch Functions.

Emulating z/OS JCL Logic and Architecture

Oracle Tuxedo Application Runtime for Batch provides a set of high-level functions that simplify script syntax enabling more readable and more easily maintainable Korn shell scripts.

Using these functions ensures consistent services; when used together, execution of one function can be conditional on the value of the return code produced by a preceding function.

A function is generally called directly from a Korn shell script resulting from JCL conversion.

Oracle Tuxedo Application Runtime for Batch normalizes Korn shell script formats by proposing a script model where the different execution phases of a job are clearly identified.

EJR syntax

Synopsis

```
EJR [-h]

EJR [-v] [-d regexp] [-f EnvFile] [-t file] [-V n] Job [parameters]
```

Arguments

-h
help — displays the command syntax. Reserved for maintenance team.

-d regexp
Debug mode — with a regular expression (regexp) describing the functions to debug, for example `-d "m_FileLoad"` to debug the `m_FileLoad` function.

-v
Verbose mode — the execution report is displayed on screen during execution. By default not activated.

-f EnvFile
Specifies an environment file that is different from the default (BatchRT.conf) file.

-t file
Test mode — this option runs the script without executing the different steps. It allows to check the kinematics of the Korn shell script. Reserved for maintenance team.

-V n
Level mode (0 to 9).

Job
The job name — the name of the script to be launched without the `.ksh` extension.

Parameters
Optional parameters for the job.

Tools for managing the execution of jobs

Log file management

When a script is launched with EJR, a log file is generated. When not using TuxJES, the name of the log file is:

`JobName_YYYYMMDDHHMMSS_Jobid.log.`

The log file is created in a directory identified by the MT_LOG environment variable. The contents of this file provide the production team with detailed information about the execution of a job.

When using TuxJES, refer to the related documentation.

Return code management

Oracle Tuxedo Application Runtime for Batch uses several return-code variables to manage the result of a function execution and the result of job execution.

Table 3-1 Oracle Tuxedo Application Runtime for Batch Return Codes

Return Code	Description
MT_RC	The Return code for an Oracle Tuxedo Application Runtime for Batch function execution. • If MT_RC < MT_RC_ABORT: the function has terminated successfully. • If MT_RC >= MT_RC_ABORT there is an execution error, the return code will contain a UNIX command return code or a specific value depending on the origin of the error.
MT_RC_JOB	General return code (for the job) MT_RC_JOB is updated with MT_RC at the end of each phase. It contains the maximum MT_RC value for the job.
MT_RC_ABORT	Minimum return code value to consider the job is aborted. This value depends on the source machine and application. Default value is 1.
MT_RC_STEP_RETURNCODE_{LABEL}	Each phase return code is saved. The variable name contains the phase label. This variable can be used for specific chaining within the script.

Cobol runtime

A COBOL runtime, `runb`, is provided to initialize the execution context of a user Cobol program before the call for its execution. This runtime is used instead of the standard Cobol runtime.

Oracle Tuxedo Application Runtime for Batch Purpose

- Abort trapping procedure definition (standard: std_proc_error and database: dba_proc_error). The standard procedure traps Cobol errors and traces them in a log file. The Database procedure executes a rollback function to insure data integrity.
- Database access function tracing management (mw_dbstat).
- Database Connection and Disconnection and data integrity control (COMMIT and ROLLBACK) if the program is run (m_ProgramExec) with -b option.
- Cobol program execution.

DataBase interaction management

Oracle Tuxedo Application Runtime for Batch takes care of the Database context usage:

- Initialization: If a Cobol program is executed (m_ProgramExec) with the -b option, the runtime (runb) connects it to the database according to the values of environment variables. The variable MT_DB_LOGIN must have a correct value (user name, password and Oracle instance, at least "/").
- Termination: Depending on the program return code, the Batch Runtime executes a COMMIT (MT_RC_JOB = 0) or a ROLLBACK (MT_RC_JOB != 0), then disconnects from Database.

Testing the Validity of a Script (non-exec mode)

Tip: This feature is reserved for the maintenance team.

Using the `-t` argument, it is possible to run the KSH script without executing the internal functions. The `-t` argument allows a script to be checked (for example a newly-developed script) and verify the chaining of the different phases.

Oracle Tuxedo Application Runtime for Batch Functions

Naming Convention

The names of the Batch Runtime functions respect the following format:

`prefix_ObjectAction`

Where:

prefix_

m
: specifies an external function.

mi
: specifies an internal function.

Object
: is the type of object on which the function is used and

Action
: is the action to be executed on the object.

Examples include:

- **m_FileAssign**
- **m_FileBuild**
- **m_RcTest**
- **m_ProgramExec**

Reference Page Command Syntax

Unless otherwise noted, commands described in the Synopsis section of a reference page accept options and other arguments according to the following syntax and should be interpreted as explained below.

name [-option . . .] [cmdarg . . .]
: where **name** is the name of an executable file and `option` is a string of one of the following two types: **noargletter** . . . or **argletter optarg [, . . .]**

  An option is always preceded by a "-".

noargletter

  A single letter representing an option that requires no option-argument. More than one noargletter can be grouped after a "-" .

optarg

  A character string that satisfies a preceding argletter. Multiple optargs following a single argletter must be separated by commas, or separated by white space and enclosed in quotes.

cmdarg
: A pathname (or other command argument) that represents an operand of the command.

-

(dash) By itself means that additional arguments are provided in the standard input.

--

(two dashes) Means that what follows are arguments for a subordinate program.

[ ]

Surrounding an option or cmdarg, mean that the option or argument is not required.

{ }

Surrounding cmdargs that are separated by an or sign, mean that one of the choices must be selected if the associated option is used.

"OR" argument

. . .

Means that multiple occurrences of the option or cmdarg are permitted.

Reference

The Oracle Tuxedo Application Runtime for Batch Reference Guide describes, in alphabetic order, shell-level functions delivered with the Batch Runtime software.

The following functions are described:

Table 3-2

Functions		
m_CondElse	m_FileEmpty	m_OutputAssign
m_CondEnd	m_FileExist	m_OutputSet
m_CondExec	m_FileLoad	m_PhaseBegin
m_CondIf	m_FileOverride	m_PhaseEnd
m_DirCopy	m_FileSort	m_ProcBegin
m_DirCreate	m_FileRename	m_ProcEnd
m_DirDelete	m_GenCommit	m_ProcInclude
m_DirRename	m_GenDefine	m_ProgramExec
m_ExecSQL	m_GenRollback	m_RcSet
m_FileAssign	m_JclLibSet	m_ShellInclude
m_FileBuild	m_JobBegin	m_StepLibSet
m_FileClrData	m_JobEnd	m_SymbolDefault
m_FileDelete	m_JobLibSet	m_SymbolSet

Overview

The functions correspond to the interface (API) between the shell script and the Batch Runtime executable. Some scripts, such as m_JclibSet, are used only in the conversion stage and are not present in the extended script that is available for execution.

m_CondElse

Name

m_CondElse — Else of a condition.

Synopsis

```
m_CondElse
```

Description

This function marks the alternative part of a m_CondIf function.

Options

No parameters.

m_CondEnd

Name

m_CondEnd - End of a condition

Synopsis

```
m_CondEnd
```

Description

This function ends the previous IF condition.

Options

No parameters.

m_CondExec

Name

m_CondExec — conditional execution (for a program or procedure).

Synopsis

```
m_CondExec condexp [condexp...]
```

Description

Conditional execution. If condition is true, the remaining command in the current step is ignored. Each condition expression contains either, EVEN, ONLY or a value, operator[,step] condition. An m_CondExec statement may contain several condition expressions and this association specifies a logical "OR" of the different conditions.

Options

Condexp [condexp]

Condition expression.

EVEN
: Executes step even if previous step ended abnormally.

ONLY
: Executes step only if previous step ended abnormally.

value, operator[,step]
: Where <step> is any of the previous steps. If the previous step was not executed, the condition is false.

Examples

```
m_CondExec EVEN
```

```
m_CondExec 4,LT,STEPEC01 8,LT,STEPEC02 ONLY
```

m_CondIf

Name

m_CondIf - Conditional execution

Synopsis

```
m_CondIf "condexp [condexp…]"
```

Description

Executes the condition contained in the " condexp " parameter. Nested levels of "if" conditions are authorized.

Options

"Condexp [condexp]"

Conditional expression.

RC,operator,value

`RC` indicates a return code.

STEP.RC,operator,value

`STEP.RC` indicates that the expression tests a return code for a specific STEP.

`Operator` indicates the operator used for the conditional expression (GT, LT, EQ etc.).

ABEND

`ABEND` indicates an abend condition occurred.

ABENDCC=number

`ABENDCC` indicates a system or user completion code.

Examples

```
m_CondIf " RC,EQ,3"
```

Note: The statements following this m_CondIf statement are executed if the return code is equal to 3.

m_DirCopy

Name

m_DirCopy – Copies the members of a directory.

Synopsis

```
m_DirCopy InDir OutDir
```

Description

This function copies the members of a directory to another directory.

Options

InDir

The name of the directory to be copied.

OutDir

The name of the output directory.

Example

```
m_DirCopy ${DATA}/PJ01DDD.BT.DATA.PDSA ${DATA}/PJ01DDD.BT.DATA.PDSE
```

m_DirCreate

Name

m_DirCreate – Creates a directory.

Synopsis

```
m_DirCreate DirName
```

Description

This function creates a directory.

Options

DirName

The name of the directory to be created.

m_DirDelete

Name

m_DirDelete – Deletes a directory.

Synopsis

```
m_DirDelete DirName
```

Description

This function deletes a directory.

Options

DirName

The name of the directory to be deleted.

m_DirRename

Name

m_DirRename – Renames a directory.

Synopsis

```
m_DirRename DirName
```

Description

This function renames a directory.

Options

DirName

The name of the directory to be renamed

m_ExecSQL

Name

m_ExecSQL — Executes an SQL script.

Synopsis

```
m_ExecSQL -f <sysin file>
```

Description

This function executes an SQL script.

Options

-f <sysin file>

The <sysin file> contains the SQL directives (CREATE TABLE, CREATE INDEX, DELETE, SELECT …)

Example

```
cat <<_end >${TMP}/SYSIN_STEPDB04_${MT_JOB_NAME}_${MT_JOB_PID}

SELECT * FROM PJ01DB2.TABTEST2;

_end

m_FileAssign -d OLD SYSIN

${TMP}/SYSIN_STEPDB04_${MT_JOB_NAME}_${MT_JOB_PID}

m_ExecSQL -f ${DD_SYSIN}
```

Note: The DB2 commands are not translated. The user has to verify these commands according to the target data base software.

m_FileAssign

Name

m_FileAssign — Assigns a file.

Synopsis

```
m_FileAssign -c [-d] [-g] ddname dsname
```

Description

`m_FileAssign` assigns a file. If assigning a file triggers the creation of a file, the creation process precedes the assign itself.

Specific cases are:

- New files (`DISP=NEW` parameter).
- Concatenated files (DD cards, where only the first one contains a label). In this case a concatenation is made in a temporary file, the original DSNAME is replaced by the name of the temporary file.
- Override files (file override in the JCL); a specific assign function m_FileOverride is used. This function call is implanted in each STEP required, before the execution of the program.
- In the case where a file assign contains a `DISP=NEW,DELETE,DELETE` parameter, a delete process is added to the end (normal and abnormal) of the step.
- For the `DISP=OLD` and `DISP=PASS` options, the file is kept.
- For the `DISP=MOD` option, the write to the file is made in a temporary intermediary file, then by a copy in `Extend` on the original file.

Options

`-c <concatenation>`

Concatenate this file with the previous `dsname` for this `ddname`.

`-d <DispOption>`

This option indicates the DISPosition status of the file in the format:

DISP=([status][,normal-termination-disp][,abnormal-termination-disp])

Possible combinations are:

```
DISP= (    [NEW]    [,DELETE ]    [,DELETE ] )
           [OLD]    [,KEEP]       [,KEEP ]
           [SHR]    [,PASS]
           [MOD]
```

The Disp Option indicates the status of the data set at the beginning of a job step and what to do with the data set in the event of normal and abnormal termination of the step.

<status>
The status indicates if an existing data set should be used or a new one created. For existing data sets the status indicates if the data set can be shared with other jobs or used to append records to the end of the data set. the possible values are:
NEW — indicates to create a new unshared data set.
OLD — indicates to use an existing unshared data set.
SHR — indicates to use an existing shared data set.
MOD — indicates an existing unshared data set to add records at the end of file.

<normal-termination-disp>
This option indicates what to do with a data set when a step ends normally. The possible values are:
DELETE — The data set is no longer needed.
KEEP — The data set is to be kept.
PASS — The data set is to be passed for use by a subsequent step.

(CATLG is not supported).

<abnormal-termination-disp>
DELETE — The data set is no longer needed.
KEEP — The data set is to be kept.

(CATLG is not supported).

The termination dispositions have default values for each status, the default values are:
NEW: DELETE, DELETE
OLD/SHR/MOD : KEEP, KEEP

Note: PASS is functionally equivalent to KEEP.

`-g <generationFile>`

Indicates that the data set is a generation file.

`ddname <InternalFileName>`

The logical name of the file as defined in the SELECT statement of the COBOL program.

`dsname <ExternalFileName>`

Real file name, full path of the file on the disk.

Examples

```
m_FileAssign -d SHR ENTREE ${DATA}/PJ01DDD.BT.QSAM.KBIEI001
```

m_FileBuild

Name

m_FileBuild — Creates a file.

Synopsis

```
m_FileBuild -t <type> -r <record length> [-k <primary key] [-K secondary
key] >filename>
```

Description

This function creates a file.

Options

-t Type

Type is the organization type of the created file. The possible values are:

SEQ
: for a sequential file.

LSEQ
: for a line sequential file.

INDX
: for an indexed file

Note: The options -r and -k are mandatory for an indexed file.

-r Length

Indicates the record length of the file. This option is mandatory for indexed files.

-k Position+Length

The primary key (mandatory for indexed files)

Position
: The first character of the key in relation to the beginning of the record.

Length
: The length of the primary key.

-K Position+Length[d] [Position+Length[d] ...]

The secondary key indicating that the file contains one or more secondary keys. Each secondary key may permit duplicates.

`Position`
: The first character of the key in relation to the beginning of the record.

`Length`
: The length of the primary key.

`d`
: Permit duplicates of secondary key.

Examples

To build an indexed file with no secondary key, the following function builds an indexed file with a record length of 266 bytes. There is no secondary key and the primary key begins in the first character of the record and is six characters long

```
m_FileBuild -t IDX -r 266 -k 1+6 ${DATA}/METAW00.VSAM.CUSTOMER
```

To build a similar indexed file, with in addition, a non-duplicate secondary key in position 20 with a length of 7 the following function can be used:

```
m_FileBuild -t IDX -r 266 -k 1+6  -K 20+7 ${DATA}/METAW00.VSAM.CUSTOMER
```

To build a similar indexed file with a secondary key allowing duplicates in position 20 with a length of 7 the following function can be used:

```
m_FileBuild -t IDX -r 266 -k 1+6  -K 20+7d
${DATA}/METAW00.VSAM.CUSTOMER
```

m_FileClrData

Name

m_FileClrData - clears a file.

Synopsis

```
m_FileClrData FileName
```

Description

`m_FileClrData` is used to clear a file.

Options

FileName
The name of the file to be cleared.

Example

```
m_FileClrData ${DATA}/PJ01DDD.BT.QSAM.KBSTO045
```

m_FileDelete

Name

m_FileDelete — Deletes a file.

Synopsis

```
m_FileDelete FileName
```

Description

`m_FileDelete` is used to delete a a file.

Options

FileName

The name of the file to be deleted.

Example

```
m_FileDelete ${DATA}/PJ01DDD.BT.QSAM.KBSTO045
```

m_FileEmpty

Name

m_FileEmpty – Checks whether a file is empty.

Synopsis

```
m_FileEmpty -r ReturnVariable FileName
```

Description

`m_FileEmpty` is used to check whether a file is empty.

Options

-r ReturnVariable
Returns “true” or “false”.

FileName
The name of the file to be checked.

m_FileExist

Name

m_FileExist – Checks the presence of a file.

Synopsis

```
m_FileExist –r ReturnVariable FileName
```

Description

`m_FileExist` is used to check whether a file is present.

Options

-r ReturnVariable
Returns “true” or “false”.

FileName
The name of the file to be checked.

m_FileLoad

Name

m_FileLoad — Loads a file.

Synopsis

```
m_FileLoad [-C] [-S] Infile [Infile ...] Outfile
```

Description

This function loads a file.

Options

-C

Number of records to copy from the `Infile` to the `Outfile`.

-S

Number of records to skip when copying from the `Infile` to the `Outfile`.

Example

```
m_FileLoad ${DD_SYSUT1} ${DD_SYSUT2}
```

m_FileOverride

Name

m_FileOverride — Overrides a file.

Synopsis

```
m_FileOverride [-d] -S label ddname dsname
```

Description

`m_FileOverride` overrides a file assignment, this assign has priority over a standard assign (m_FileAssign).

Options

See m_FileAssign for `d`, `ddname` and `dsname` parameters.

-S

Name of the label in the called procedure.

Example

```
m_FileOverride -d OLD -s PR3STEP1 SYSIN

{TMP}/SYSIN_PR3STEP1_${MT_JOB_NAME}_${MT_JOB_PID}
```

m_FileSort

Name

m_FileSort — Sorts a file.

Synopsis

```
m_FileSort -s SortSpecificationFile |
--specification_file=SortSpecificationFile Infile [Infile ...] [Outfile]
```

Description

This function sorts a file.

Options

-s

The sort specification indicates either a file containing the sort specification or a file that indicates where the sort specification is to be found.

Infile

At least one file must be used as input to the sort.

Outfile

Not mandatory when the -n option is used.

m_FileRename

Name

m_FileRename – Renames a file.

Synopsis

```
m_FileRename OldName NewName
```

Description

`m_FileRename` is used to rename a file.

Options

NewName
The new name of the file.

OldName
The old name of the file.

m_GenCommit

Name

m_GenCommit — Commits a generation file.

Synopsis

```
m_GenCommit [GDG base name]
```

Description

`m_GenCommit` commits a generation file.

Options

`GDG base name`
: Name of the generation file to commit.

m_GenDefine

Name

m_GenDefine — Defines a GDG base name.

Synopsis

```
m_GenDefine -s --nb_occurs <GDG base name>
```

Description

This function defines a GDG.

Options

-s

Number of occurrences of generation file to keep on disk.

GDG base name

The name of the `GDG base` for which the maximum number of generations is being defined.

Example

```
m_GenDefine -s 31 ${DATA}/PJ01DDD.BT.GDG.KBIDU001
```

m_GenRollback

Name

m_GenRollback — Rolls back a generation GDG base name.

Synopsis

```
m_GenRollback [GDG base name]
```

Description

This function rolls back a GDG base name

Options

GDG base name

Name of the generation file to rollback.

m_JclLibSet

Name

m_JclLibSet — Specify conversion stage Procedure and Include directories.

Synopsis

```
m_JclLibSet directory
```

Description

m_JclibSet specifies the directories where Procedures and Includes are stored during the conversion phase.

Options

directory
Path and name of the directory.

Example

```
m_JclLibSet PJ01DDD.BT.INCLUDE.SRC
```

m_JobBegin

Name

m_JobBegin — Used to launch a job.

Synopsis

```
m_JobBegin -j jobname [-c class] [-p priority] [-r restart] [-t typrun] -v
version -s start_label
```

Description

Indicates the parameters that are used on the z/OS job card with the JES2 interface. The parameters are stored in the following files:

- *class* is stored in the *JOBID*.class file
- *restart* is stored in the *JOBID*.restart file
- *priority* is stored in the *JOBID*.priority file
- *typrun* is stored in the *JOBID*.typrun file

Options

-j *jobname*
: The name of the job to launch.

-c *class*
: The execution class of the job.

-p *priority*
: The execution priority of the job.

-r *restart*
: The name of the step to use to restart the job.

-t *typrun*
: Indicates what should be done with the job. One of the following choices:

  COPY – Copy the job directly in an output stream to sysout.

  HOLD – The system should hold the job.

  JCLHOLD – JES2 should hold the job.

  SCAN – Scan JCL for syntax errors only.

`-v` *`version`*
: version of the ksh script.

`-s` *`start_label`*
: Start label — label of the first phase to be started.

Example

```
m_JobBegin -j PJ01DSTA -s START -v 1.0 -t SCAN
```

m_JobEnd

Name

m_JobEnd — Ends a job.

Synopsis

```
m_JobEnd
```

Description

This function is used to end a job.

Options

```
None
```

m_JobLibSet

Name

m_JobLibSet — Specifies where programs are stored.

Synopsis

```
m_JobLibSet directory [:directory[:directory…]]
```

Description

This function specifies at job level the directory in which programs are stored.

Options

directory [:directory[:directory…]]

Path and name of the directory containing executable programs.

m_OutputAssign

Name

`m_OutputAssign` – manages DD SYSOUT statements with the following parameters: CLASS, COPIES, DEST, FORMS and HOLD.

Synopsis

```
m_OutputAssign [-c <class>] [-n <copies>] [-d <dest>] [-f <forms>]
      [-H <Y/N>] [-o <reference[,reference,..]>] [-D dsname] ddname
```

Options

-c <class>
Class of the output queue.

-n <copies>
Number of copies to print.

-d <dest>
Destination of the printing.

-f <forms>
Name of the used form

-H<Y/N>
Specifies whether the print must held or not.

N is the default value.

-o <reference[,reference,…]>
List of " OUTPUT " references.

-D <dsname>
Data set name.

ddname
Data Definition Name

m_OutputSet

Name

`m_OutputSet` — manages the "OUTPUT JCL" statement with the following parameters: CLASS, COPIES, DEST, FORMS and PRIORITY.

Synopsis

```
m_OutputSet [-c <class>] [-n <copies>] [-d <dest>] [-f <forms>] [-p <prty>]
      [-D Y/N] Reference
```

Options

-c <class>
Class of the output queue.

-n <copies>
Number of copies to print.

-d <dest>
Destination of the printing.

-f <forms>
Name of the form used.

-p <priority>
Specifies the priority of the output

-D
Default reference (Y/N)

Reference
Reference name of the output.

m_PhaseBegin

Name

m_PhaseBegin — Called at the beginning of a script phase.

Synopsis

```
m_PhaseBegin
```

Description

This function is called at the beginning of a script phase.

Options

```
None.
```

m_PhaseEnd

Name

m_PhaseEnd — Called at the end of a script phase.

Synopsis

```
m_PhaseEnd
```

Description

This function is called at the end of a script phase.

Options

```
None.
```

m_ProcBegin

Name

m_ProcBegin — Begins an in-stream procedure.

Synopsis

```
m_ProcBegin ProcedureName
```

Description

An in-stream procedure is added at the end of a korn shell script (by Oracle Tuxedo Application Runtime WorkBench during the translation) and referenced by `m_ProcInclude`.

Options

`ProcedureName`
Name of the procedure to include.

Example

```
m_ProcBegin  KBPRB007
```

m_ProcEnd

Name

m_ProcEnd — Ends an in-stream procedure.

Synopsis

```
m_ProcEnd
```

Description

An in-stream procedure added at the end of a korn shell script is ended by m_ProcEnd.

Options

None.

m_ProcInclude

Name

m_ProcInclude — Calls a procedure to be included in the script during the conversion phase.

Synopsis

```
m_ProcInclude ProcedureName
```

Description

Options

ProcedureName
: Name of the (in-stream or catalogued) procedure to include.

Example

```
m_ProcInclude BPRAP001
```

m_ProgramExec

Name

m_ProgramExec — Executes a program.

Synopsis

```
m_ProgramExec [-b] [-e exit_type] Program [arguments]
```

Description

This function runs a COBOL program.

Options

`-b`

Indicates the database will be accessed by the program.

`-e`

Indicates an exit routine should be used, an exit routine may be a begin or an end user routine. An exit type can be indicated either BOTH, BEGIN, or END in the exit routine name: "RTEX_"exitName"_Begin" ans/or "RTEX_"exitName"_End"

`Program [arguments]`

User arguments to be passed to the program.

Examples

```
m_ProgramExec BPRAB006 "08"
```

Indicates to run program BPRA006 with the parameter "08"

```
m_ProgramExec -b BDBAB001
```

Indicates that the program BDBAB001 accesses the Data Base

Note: To pass a parameter to a program

The <"> (double quote) character is used to mark out the boundaries of the parameters

Examples:

`PARM=MT5` on z/OS becomes `"MT5"` on target

`PARM=(MT5,MT6)` on z/OS becomes `"MT5,MT6"` on target

`PARM='S=MT5'` on z/OS becomes `"S=MT5"` on target

PARM=('S=MT5','Q=MT6') on z/OS becomes "S=MT5,Q=MT6" on target

Two successive <''> (2 simple quotes) are replaced on one <'> (1 simple quote).

`PARM='5 O''CLOCK'` becomes `"5 O'CLOCK"`

Two successive `&&` (2 ampersands) are replaced by one `&` (1 ampersnd) character.

`'&&TEMP'` becomes `"&TEMP"`

m_RcSet

m_RcSet <ARGS> ReturnCode [Message]

Name

m_RcSet — Sets the return code.

Synopsis

```
m_RcSet ReturnCode [Message]
```

Description

m_RcTest sets the return code of a function.

Options

`ReturnCode`
: The value of the return code of the current phase.

`Message`
: A message that may be displayed with the return code.

Examples

```
m_RcSet ${MT_RC_ABORT:-S999} "Unknown label : ${CURRENT_LABEL}"

m_RcSet 0
```

m_ShellInclude

Name

m_ShellInclude — Inserts a part of script.

Synopsis

```
m_ShellInclude script name
```

Description

This function inserts a part of script.

Options

script name

Name of the part of a script to be included in the script shell during the conversion phase.

m_StepLibSet

Name

m_StepLibSet — Specifies where programs are stored.

Synopsis

```
m_StepLibSet directory [:directory[:directory…]]
```

Description

m_SteplibSet specifies at step level where programs are stored. This information is interpreted when the program is to be executed.

Options

directory

Path and name of the directory containing executable programs.

m_SymbolDefault

Name

`m_SymbolDefault` — Assigns a value to a symbol.

Synopsis

```
m_SymbolDefault var=value
```

Description

Used before the call of a procedure to define default substitution texts for symbols in the procedure.

This function will be analyzed and taken into account during the conversion phase and the symbols replaced by their value in the extended script.

Options

var
: Name of the variable.

Value
: Value assigned to the variable.

Example

```
m_SymbolDefault VAR=45
```

m_SymbolSet

Name

m_SymbolSet — Defines a symbol.

Synopsis

```
m_SymbolSet var=value
```

Description

Defines a symbol and assigns a value before the first use of this symbol.

Options

var

Name of the variable.

Value

Value assigned to the variable.

Example

```
m_SymbolSet VAR=45
```

CHAPTER 4

Tuxedo Job Enqueueing Service (TuxJES)

This chapter describes servers, commands and utilities included in the TuxJES feature.

Table 1 lists TuxJES commands and functions.

Table 1 TuxJES Servers, Commands and Utilities

Name	Description
genappprofile	Generates the security profile for TuxJES system
artjesadmin	TuxJES command interface.
ARTJESADM	TuxJES administration server.
ARTJESCONV	TuxJES conversion server.
ARTJESINITIATOR	TuxJES job control API.
ARTJESPURGE	Purges job queue.
TuxJES Queue System	TuxJES Queue system.

genappprofile

Name

genappprofile – Generates the security profile for TuxJES system

Synopsis

```
genapprofile [-f userprofile]
```

Description

This utility generates the security profile for TuxJES system. When `genappprofile` is launched, you are prompted to enter the Oracle Tuxedo application password, user name, user password. The output is a security profile file which contains the user name and encrypted passwords. The generated security profile file can be used by the `artjesadmin` tool to login to an Oracle Tuxedo domain.

Parameters and Options

`genapprofile` supports the following parameters and options:

`[-f <output_file>]`
The location of the generated security profile file. If this option is not specified, the default value is `~/.tuxAppProfile`.

See Also

artjesadmin

artjesadmin

Name

`artjesadmin` – TuxJES command interface.

Synopsis

```
artjesadmin [-f security_profile]
```

Description

`artjesadmin` is the TuxJES command interface. It requires the TuxJES system must be started first.

Parameters and Options

`artjesadmin` supports the following parameters and options:

-f

The security profile file generated by `genappprofile`. The default value is `~/.tuxAppProfile`. The user name in this profile is the owner of the submitted jobs. A job without a specified owner is assigned the owner name "*".

A job with a particular owner can only be controlled by that owner. A job without a particular owner (*) can be controlled by anyone. Any user can print all jobs.

`artjesadmin` supports the following sub commands:

`submitjob(smj ) [-o='xxx'] -i scriptfile`

Submits a job to TuxJES system. The `scriptfile` parameter is the job script to be submitted.

Note: `artjesadmin` is not responsible for scriptfile propagation. It must be located on a shared file system if the conversion and execution are not on same machines. The options are as follows:

`o='xxx'`: the option string passed to EJR

`-i =scriptfile`: The script file. It can be an absolute path format or a relative path in the current working directory. It is limited to a 1023 length in an absolute path.

Once successfully invoked, the return format `Job xxx` is submitted successfully. If an error occurs, an error message is printed.

`artjesadmin` also supports direct job submission using the following format:

```
artjesadmin [-o='xxx'] -i scriptfile
```

`artjesadmin` has a return code different from zero if there is an error occurs as listed in Table 2

Table 2 Error Codes

Code	Description
`0`	No runtime error
`251`	artjesadmin it self command error returned by ARTJESADM server
`252`	JES2SUBMIT service error
`Others`	EJR none zero exit code

printjob(ptj) -n jobname | -j jobid | -c job_class |-a [-v]

Displays the existing jobs. If no option is specified, it displays all jobs. The options are as follows:

-`n jobname`: Display jobs with given job name

-`j jobid`: Display a particular job information

`-c job_class`: Display a particular class jobs information

`-a`: Display all jobs

`-v`: Verbose mode

Listing 1 printjob Output

```
> ptj -a
JOBNAME JobID    Owner    Prty C     Status
 -------- -------- -------- --------- ---------
 cjob     00000015 *           5 A     DONE
 cjob     00000016 *           5 A     DONE
 cjob     00000018 *           5 A     CONVING

total:3
success:3
```

- `JOBNAME`: The job name.
- `JobID`: The job ID generated by TuxJES system
- `Owner`: Job Owner.
- `Prty`: Job priority
- `C`: The job class.
- `Status`: Job status
  - `EXECUTING`: a job is running
  - `CONVING`: a job waiting for conversion
  - `WAITING`: a job waiting for execution.
  - `DONE`: a job finished successfully.

– FAIL: a job finished but failed
– HOLD_WAITING: a JOB is in hold state after conversion
– HOLD_CONVING: a job is in hold state without conversion
– INDOUBT: a job is in doubt state due to its initiator restarted

In verbose mode, the job detail information is displayed:

- Submit time: The submit time of the job
- Step: The current running job step. It is only applicable to running jobs.
- Type Run: The TYPRUN definition of the job.
- Machine: Only for running/done/failed jobs. It is the machine name that the job is/was running on.
- CPU usage: The user CPU usage and system CPU usage for the job execution.
- Result: Job operation result, “OK” or error message.

If no option is specified, the "-a" option is assumed.

holdjob(hj) -n job name | -j jobid | -c job_class | -a

Hold the specified jobs which are in CONVING or WAITING status. The options are as follows:

-n jobname: Hold jobs with given job name

-j jobid: Hold a particular job

-c job_class: Hold a particular class jobs

-a: Hold all jobs

If no option is specified, the "-a" option is assumed.

releasejob(rlj) -n job name |-j jobid | -c job_class | -a

Releases the jobs in HOLD_WAITING or HOLD_CONVING status so that they can be picked up by ARTJESCONV for conversion or ARTJESINITIATOR for running. The options are as follows:

-n jobname: Release jobs with given job name

-j jobid: Release a particular job

-c job_class: Release a particular class jobs

-a: Release all jobs

If no option is specified, the "-a" option is assumed.

canceljob(cj) -n job name |-j jobid | -c job_class 1 -a

Cancels a job and moves it to the output queue. For running jobs, this command informs the related ARTJESINITIATOR to invoke EJR with "-k" option. Other jobs are moved directly to the output queue. The TuxJES system assumes the job is terminated when EJR returns. The options are as follows:

-n jobname: Cancel jobs with given job name

-j jobid: Cancel a particular job

-c job_class: Cancel a particular class jobs

-a: Cancel all jobs

If no option is specified, the "-a" option is assumed.

purgejob(pgj) -n job name | -j jobid | -a

Completed jobs in the output queue are moved to the purge queue. For other jobs, purgejob has same effect as canceljob. The purgejob command does not purge the job directly. The ARTJESPURGE server deletes the job from the TuxJES system. If ARTJESPURGE is not started, the job remains in the output queue.

The options are as follows:

-n jobname: Purge jobs with given job name

-j jobid: Purge a particular job

-a: Purge all jobs

If no option is specified, the "-a" option is assumed.

event (et) [-t S,C,E,P,A] on|off

This command tells artjesadmin to subscribe particular job event. The options are:

S: job submission event; the event name is ARTJES_JOBSUBMIT

C: job conversion complete event; the event name is ARTJES_JOBCVT

E: job execution finish event; the event name is ARTJES_JOBEXEC

P: job purge event; the event name is ARTJES_ARTJESPURGE

A: all supported events. If the event is set to "on", A is the default.

on |off: The submission is on or off. the "on" setting can be used with the -t option. "off" will unsubscribe all event subscriptions.

If the subscribed event type is not configured in JESCONFIG, an error is reported.

verbose(v) on|off
Turn on /off verbose mode.

See Also

Oracle Tuxedo Application Runtime for Batch User Guide

ARTJESADM

Name

ARTJESADM – TuxJES Administration server.

Synopsis

```
ARTJESADM
SRVGRP="identifier"
SRVID="number" CLOPT=" [-A][servopts options] -- -i JESCONFIG"
```

Description

ARTJESADM is an Oracle Tuxedo application server provided by TuxJES. The artjesadmin command communicates with ARTJESADM for most tasks.

ARTJESADM must be configured in the UBBCONFIG file in front of other TuxJES servers since others they access services provided by ARTJESADM. If JESCONFIG is changed, all TuxJES related servers must be restarted for new configurations to take effect.

Parameters and Options

ARTJESADM supports the following parameters and options:

-i JESCONFIG
JESCONFIG represents the full path name of the TuxJES system configuration file. It allows the following parameters:

JESROOT
The full path name of the root directory to store job information. It is a mandatory attribute. If this directory does not exist, ARTJESADM creates it automatically.

```
JESROOT=/xxx/xxx
```

DEFAULTJOBCLASS
The default job class if the job class is not set for a job. It is an optional attribute. The default job class is A if this attribute is not set.

```
DEFAULTJOBCLASS=[A-Z],[0=9]
```

DEFAULTJOBPRIORITY

The default job priority if the job priority is not set for a job. It is an optional attribute. The default job priority is 0 if this attribute is not set.

DEFAULTJOBPRIORITY=[0-15]

DUPL_JOB=NODELAY

If it is not set, only one job can be in execution status for a job name. NODELAY will remove the dependency check. The default value is delay execution.

EVENTPOST=S,C,E,P,A

Specifies whether events are posted for a job at particular stages.

S: Job submission event. Event name: ARTJES_JOBSUBMIT

C: Job conversion complete event. Event name: ARTJES_JOBCVT

E: Job execution finish event. Event name: ARTJES_JOBFINISH

P: Job purge event. Event name:ARTJES_JOBPURGE

A: All supported events.

If EVENTPOST is not specified, no events are posted. The data buffer with event pos is FML32 type and the fields are defined in JESDIR/include/jesflds.h.

Example(s)

UBBCONFIG example:

```
ARTJESADM
        SRVID=1 SRVGRP=SYSGRP CLOPT="-A -- -i /nfs/users/jes/jesconfig"
```

See Also

Oracle Tuxedo Application Runtime for Batch User Guide

ARTJESCONV

Name

ARTJESCONV – TuxJES conversion server.

Synopsis

```
ARTJESCONV
SRVGRP="identifier"
SRVID="number" CLOPT=" [-A][servopts options] -- "
```

Description

The TuxJES conversion server. It is responsible for invoking the EJR to do the job conversion.

Example(s)

UBBCONFIG example:

```
ARTJESCONV
        SRVID=2 SRVGRP=SYSGRP CLOPT="-A -- "
```

See Also

Oracle Tuxedo Application Runtime for Batch User Guide

ARTJESINITIATOR

Name

ARTJESINITIATOR – Job Initiator

Synopsis

```
ARTJESINITIATOR
SRVGRP="identifier"
SRVID="number" CLOPT=" [-A][servopts options] -- -C jobclasses"
```

Description

ARTJESINITIATOR is an Oracle Tuxedo application server provided the TuxJES. It is responsible for invoking the EJR to execute the jobs.

Once a ARTJESINITIATOR is killed or shutdown while it has job running, it will put the job in the INDOUBT state when it is restarted.

Parameters and Options

ARTJESINITIATOR supports the following parameters and options:

-c jobclass[jobclass].
Specifies the job classes this ARTJESINITIATOR server is associated. If this option is not specified, ARTJESINITIATOR fails to start.

Example(s)

UBBCONFIG example:

```
ARTJESINITIATOR
```

```
SRVID=3 SRVGRP=SYSGRP MIN=10 CLOPT="-A -- -c AHZ"
```

In this example, ten ARTJESINITIATOR instances are configured and are associated with the "A","H" and "Z" job classes.

See Also

Oracle Tuxedo Application Runtime for Batch User Guide

ARTJESPURGE

Name

ARTJESPURGE – Purges job queue

Synopsis

```
ARTJESPURGE
SRVGRP="identifier"
SRVID="number" CLOPT=" [-A][servopts options] -- "
```

Description

ARTJESPURGE monitors the purge queue. If it finds a job in the purge queue, it removes the job in the queue and deletes the directory JESROOT/<JOBID>.

See Also

Oracle Tuxedo Application Runtime for Batch User Guide

TuxJES Queue System

In order to emulate the z/OS JES2 system, TuxJES system uses a queue mechanism for batch job life cycle management. All queues are created in one queue space called "`JES2QSPACE`". A batch job is represented by a message that resides and is transferred to queues listed in Table 3.

Table 3 TuxJES **Queues**

Queues	Description
Conversion Queue	When a batch job is submitted to the TuxJES system, it is put in the conversion queue first. There is only one conversion queue in the system. A converted job is moved from the "conversion queue" to the "execution queue". The jobs in the queue are processed in FIFO order. Queue name: CONV
Hold Queue	If a job is in the HOLD state (JCLHOLD or HOLD), it is put in the hold queue. Once released, it is moved to the conversion queue or waiting queue based on the `typrun` parameter. Queue name: HOLD
Execution Queue	There are 36 job classes (A-Z and 0-9). A job also has a priority value ranging from 0 to 15. The jobs are scheduled based on the job class and priority. One job class is mapped to one /Q queue, (36 queues all together). These are the queues where the job is stored staying and waits for execution. The job priority is mapped to the queue message priority. All queues are created based on priority. Queue names: [A-Z], [0-9].
Executing Queue	This queue stores running/executing jobs. There is only one "executing queue" in the system. When a job is picked up from an "execution queue" and successfully goes to running state, the job is moved to the "executing queue". The jobs in this queue are processed in FIFO order. Queue name: EXEC
Output Queue	When a job is completed or an error occurs, it is sent to the "output queue". There is only one "output queue". The jobs in the queue are processed in FIFO order. Queue name: OUTPUT

Table 3 TuxJES **Queues**

Queues	Description
Purge Queue	When a job is to be purged, it is moved to the purge queue. There is only one "purge queue" in the system. Queue name: PURGE
Internal Queues	ART JES also has some internal queues on the JES2QSPACE for internal usage.

The TuxJES Queue Creation Script

The TuxJES system provides a sample shell script (`jesqinit`) to create the queue space (`JES2QSPACE`) and the queues listed in Table 3. You can modify the script to adapt to your environment, but must adhere to the following:

1. Queue order can not be changed
2. Fixed queue names and queue space name
3. The script can be customized for queue space/queue creation parameters

Recommended /Q creation Values

Device Size of Pages: 10000

Queue Space Size of Pages: 5000 (We assume the max number of jobs is 10000, each job will consume 2k bytes and the page size is 4k)

Number of Messages in Queue: 10000

Number of Concurrent Transactions: 1000

Number of Concurrent processes in queue: 100

Note: These parameters can be customized according to the specific environment.