Oracle Tuxedo Application Runtime for Batch

Reference Guide

11*g* Release 1 (11.1.1.2)

July 2011

Contents

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 statements 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:

In the column "status", the following abbreviations are used

N.R. means "not relevant"

N.S. means "not supported"

JCL Card Equivalence Table

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
//*		Comments.	supported
/*		In-stream data delimiter	supported
//		End of job	supported
COMMAND			N.S.
CNTL/ENDCNTL			N.R.
DD	*	**m_FileAssign -i**	supported
	ACCODE		N.R.
	AMP		N.R.
	AVGREC		N.R.
	BLKSIZE		N.R.
	BLKSZLIM		N.R.
	CCSID		N.R.
	CHKPT		N.S.
	CNTL		N.R.
	DATA	**m_FileAssign**	supported
	DATACLAS	**see DATACLAS chapter**	supported
	DCB	**see DCB chapter**	supported
	DISP	**m_FileAssign -d <DISP option>**	supported
	DLM	**m_FileAssign -D <delimiter>**	supported
	DSID		N.R.
	DSNAME	**m_FileAssign**	supported

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
DD (continued)	DSNTYPE		N.S.
	DUMMY	**m_FileAssign with /dev/null**	supported
	DYNAM	**m_FileAssign with /dev/null**	supported
	EXPDT		N.R.
	FILEDATA		N.R.
	FREE		N.R.
	KEYLEN	**m_FileAssign -k**	supported
	KEYOFF	**m_FileAssign -k**	supported
	LABEL		N.R.
	LGSTREAM		N.R.
	LIKE	**m_FileAssign -s**	supported
	LRECL	**m_FileAssign -r**	supported
	MGMTCLAS		N.R.
	MSVGP		N.R.
	PATH		N.R.
	PATHDISP		N.R.
	PATHMODE		N.R.
	PATHOPTS		N.R.
	PROTECT		N.R.
	QNAME		N.R.
	RECFM	**m_FileAssign -r**	supported
	RECORG	**m_FileAssign -T**	supported
	REFDD		N.S.

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
DD (continued)	RETPD		N.S.
	RLS		N.R.
	SECMODEL		N.R.
	SPACE		N.R.
	STORCLAS		N.R.
	SUBSYS		N.S.
	TERM		N.R.
	UNIT		N.R.
	VOLUME		N.R.

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
DD : Printing parameters	BURST		N.S.
	CCSID		N.R.
	CHARS		N.S.
	COPIES	**m_OutputAssign -c**	supported
	DEST	**m_OutputAssign -d**	supported
	FCB		N.S.
	FLASH		N.S.
	HOLD	**m_OutputAssign -H**	supported
	MODIFY		N.S.
	OUTLIM		N.S.
	OUTPUT	**m_OutputAssign -o**	supported
	SEGMENT		N.R.
	SPIN		N.S.
	SYSOUT	**see SYSOUT chapter**	supported
	UCS		N.S.
DD: Special statements	JOBLIB	**m_JobLibset**	supported
	STEPLIB	**m_StepLibset**	supported
	SYSABEND		N.S.
	SYSMDUMP		N.S.
	SYSUDUMP		N.S.
	SYSCHK		N.S.
	SYSIN	**m_FileAssign**	supported
	SYSCKEOV		N.S.

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
EXEC	ACCT		N.S.
	ADDRSPC		N.R.
	CCSID		N.R.
	COND	**m_CondExec**	supported
	DPRTY		N.R.
	DYNAMNBR		N.R.
	MEMLIMIT		N.R.
	PARM	**m_ProgramExec**	supported
	PERFORM		N.R.
	PGM	**m_ProgramExec**	supported
	PROC		supported
	RD		N.S.
	REGION		N.R.
	RLSTMOUT		N.R.
	TIME		N.S.

IF
THEN
ELSE
END
		m_CondIf
		
		m_CondElse
		m_CondEndif
			supported

INCLUDE	MEMBER	**m_ShellInclude**	
JCLLIB	ORDER	**m_JclLibSet**	

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
JOB	<jobname>	**m_JobBegin**	supported
	<accounting>		N.S.
	<prog. name>		N.S.
	ADDRSPC		N.R.
	BYTES		N.S.
	CARDS		N.R.
	CCSID		N.S.
	CLASS	**m_JobBegin -c** (with TuxJES).	supported
	COND	**m_JobBegin -C**	supported
	GROUP		N.R.
	JESLOG		N.S.
	LINES		N.S.
	MEMLIMIT		N.R.
	MSGCLASS		N.S.
	MSGLEVEL		N.S.
	NOTIFY		N.S.
	PAGES		N.S.
	PASSWORD		N.R.
	PERFORM		N.R.
	PRTY	**m_JobBegin -p** (with TuxJES).	supported
	RD		N.S.
	REGION		N.R.
	RESTART	**see RESTART chapter**	Partially supported

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
JOB (continued)	SECLABEL		N.R.
	SCHENV		N.R.
	TIME		N.S.
	TYPRUN	**m_JobBegin -t** (with TuxJES)	supported
	USER		N.S.
OUTPUT	ADDRESS		N.S.
	AFPSTATS		N.S.
	BUILDING		N.S.
	BURST		N.S.
	CHARS		N.S.
	CKPTLINE		N.S.
	CKPTPAGE		N.S.
	CKPTSEC		N.S.
	CLASS	**m_OutputSet -c**	supported
	COLORMAP		N.S.
	COMPACT		N.S.
	COMSETUP		N.S.
	CONTROL		N.S.
	COPIES	**m_OutputSet -n**	supported
	DATACK		N.S.
	DEFAULT	**m_OutputSet -D**	supported
	DEPT		N.S.
	DEST	**m_OutputSet -d**	supported

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
OUTPUT (continued)	DPAGELBL		N.S.
	DUPLEX		N.S.
	FCB		N.S.
	FLASH		N.S.
	FORMDEF		N.S.
	FORMLEN		N.S.
	FORMS	**m_OutputSet -f**	supported
	FSSDATA		N.S.
	GROUPID		N.S.
	INDEX		N.S.
	INTRAY		N.S.
	JESDS		N.S.
	LINDEX		N.S.
	LINECT		N.S.
	MAILBCC		N.S.
	MAILCC		N.S.
	MAILFILE		N.S.
	MAILFROM		N.S.
	MAILTO		N.S.
	MODIFY		N.S.
	NAME		N.S.
	NOTIFY		N.S.
	OFFSETXB		N.S.

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
OUTPUT (continued)	OFFSETXF		N.S.
	OFFSETYB		N.S.
	OFFSETYF		N.S.
	OUTBIN		N.S.
	OUTDISP		N.S.
	OVERLAYB		N.S.
	OVERLAYF		N.S.
	OVFL		N.S.
	PAGEDEF		N.S.
	PIMSG		N.S.
	PORTNO		N.S.
	PRMODE		N.S.
	PRTATTRS		N.S.
	PRTERROR		N.S.
	PRTOPTNS		N.S.
	PRTQUEUE		N.S.
	PRTY	**m_OutputSet -p**	supported
	REPLYTO		N.S.
	RESFMT		N.S.
	RETAINF		N.S.
	RETAINS		N.S.
	RETRYL		N.S.
	RETRYT		N.S.

Table 2-1 JCL Card Equivalences

JCL Card	Parameter	Equivalent in Target Environment	Status
OUTPUT (continued)	ROOM		N.S.
	SYSAREA		N.S.
	THRESHLD		N.S.
	TITLE		N.S.
	TRC		N.S.
	UCS		N.S.
	USERDATA		N.S.
	USERLIB		N.S.
	USERPATH		N.S.
	WRITER	**m_OutputSet -w**	supported
PROC		**m_ProcInclude**	supported
in-stream PROC		**m_ProcBegin**	supported
in-stream PEND		**m_ProcEnd**	supported
SET		**m_SymbolSet**	supported
XMIT			N.S.

DCB parameter of DD statement

Table 2-2 DCB parameter of DD statement

DCB parameter	Equivalent in Target Environment	Status
BFANL		N.R.
BFTEK		N.R.
BLKSIZE		N.R.
BUFIN		N.R.
BUFL		N.R.
BUFMAX		N.R.
BUFNO		N.R.
BUFOFF		N.R.
BUFOUT		N.R.
BUFSIZE		N.R.
CPRI		N.R.
CYLOFL		N.R.
DEN		N.R.
DIAGNS		N.R.
DSORG	**m_FileAssign -T**	supported
EROPT		N.R.
FUNC		N.R.
GNCP		N.R.
INTVL		N.R.
IPLTXID		N.R.
KEYLEN	**m_FileAssign -k**	supported

Table 2-2 DCB parameter of DD statement

DCB parameter	Equivalent in Target Environment	Status
LIMCT		N.R.
LRECL	**m_FileAssign -r**	supported
MODE		N.R.
NCP		N.R.
NTM		N.R.
OPTCD		N.R.
PCI		N.R.
PRTSP		N.R.
RECFM	**m_FileAssign -r**	supported
RESERVE		N.R.
RKP	**m_FileAssign -T**	supported
STACK		N.R.
TRESH		N.R.
TRTCH		N.R.

SYSOUT parameter of DD statement

Table 2-3 SYSOUT parameter of DD statement

SYSOUT parameter	Usage	Equivalent in Target Environment	Status
First parameter	Class	**m_OutputAssign -c**	supported
Second parameter	INTRDR	**m_OutputAssign -w**	supported
	SMTP	**m_OutputAssign -w**	supported
	<writer>	**m_OutputAssign -w**	supported
Third parameter	Form name	**m_OutputAssign -f**	supported
	Code name		N.S.

DATACLAS parameter of DD statement

The DATACLAS parameter is taken into account at WB JCL level only.

Using the DATACLASS informations, extracted from z/OS, the WB JCL updates the m_FileAssign function (or m_FileBuild from DATACLAS parameter in IDCAMS DEFINE commands).

RESTART parameter of JOB statement

Only the Deferred Step Restart type is managed.

The other types, Automatic Restart (Step or Checkpoint) and Deferred Checkpoint Restart, are not managed.

A deferred step restart of a job is caused by coding the function `m_JobBegin` with the `-r` parameter containing a job step and by submiting the job again.

In-Stream File

The in-stream files are supported using the `m_FileAssign` function with the `-i` parameter.

Concatenation of Files

The concatenation of files (or in-stream files) is supported using the `m_FileAssign` function with the `-C` parameter.

Override of Files

The overriding feature is supported.

Use the `m_FileOverride` function for the files managed by the `m_FileAssign` function.

Use the `m_OutputOverride` function for the files managed by the `m_OutputAssign` function.

Execution of External Sysin

Use the `m_UtilityExec` function in order to execute commands stored in a file .

General Utility Commands Equivalence Table

Table 2-4 General Utility Commands Equivalences

Utility	Command	Equivalent in Target Environment	Status
IDCAMS (modal commands)	IF/THEN/ELSE		supported
	SET		supported
	CANCEL		N.S.
	DO/END		N.S.
	MAXCC		supported
	LASTCC		supported

Table 2-4 General Utility Commands Equivalences

Utility	Command	Equivalent in Target Environment	Status
IDCAMS	ALLOCATE (main parameters)	**m_FileBuild**	supported
	ALTER NEWNAME	**m_FileRename**	supported
	ALTER LIBRARYENTRY		N.R.
	ALTER VOLUMEENTRY		N.R.
	BLDINDEX		N.R.
	CREATE LIBRARYENTRY		N.R.
	CREATE VOLUMEENTRY		N.R.
	DCOLLECT		N.S.
	DEFINE ALIAS		N.S.
	DEFINE AIX	**m_FileBuild -K**	supported
	DEFINE CLUSTER	**m_FileBuild**	supported
	DEFINE GDG	**m_GenDefine**	supported
	DEFINE NONVSAM		N.S.
	DEFINE PAGESPACE		N.R.
	DEFINE PATH		N.R.
	DEFINE USERCATALOG		N.R.
	DELETE ALAIS		N.S.
	DELETE AIX	**m_FileBuild -R**	supported
	DELETE CLUSTER	**m_FileDelete**	supported
	DELETE GDG	**m_FileDelete**	supported
	DELETE LIBRARYENTRY		N.R.
	DELETE NONVSAM	**m_FileDelete**	supported

Table 2-4 General Utility Commands Equivalences

Utility	Command	Equivalent in Target Environment	Status
IDCAMS (continued)	DELETE NVR		N.S.
	DELETE PAGESPACE		N.R.
	DELETE PATH		N.R.
	DELETE TRUENAME		N.S.
	DEL USERCATALOG		N.R.
	DELETE VOLUMEENTRY		N.R.
	DELETE VVR		N.R.
	DIAGNOSE		N.S.
	EXAMINE		N.S.
	EXPORT		N.S.
	EXPORT DISCONNECT		N.R.
	IMPORT		N.S.
	IMPORT CONNECT		N.R.
	LISTCAT		N.S.
	LISTDATA		N.S.
	PRINT	**m_FilePrint**	supported
	REPRO	**m_FileRepro**	supported
	SETCACHE		N.R.
	SHCDS		N.R.
	VERIFY		N.S.

Table 2-4 General Utility Commands Equivalences

Utility	Command	Equivalent in Target Environment	Status
IEBCOPY	ALTERMOD		N.R.
	COPY	**m_DirCopy**	supported
	COPYGRP		N.R.
	COPYMOD		N.R.
	EXCLUDE	**m_DirCopy -e**	supported
	SELECT	**m_DirCopy -s**	supported
IEBGENER	With SYSIN DUMMY	**m_FileLoad**	supported
	GENERATE MEMBER		N.S.
	GENERATE RECORD	**m_FileSort**	supported
IEFBR14		**m_ProgramExec IEFBR14**	supported
IEFBR15		**m_ProgramExec IEFBR15**	supported
PKZIP	Main parameters	**m_Pkzip**	supported
PKUNZIP	Main parameters	**m_Pkunzip**	supported
SMTP		**m_Smtp**	supported
XMITIP		**m_SendMail**	supported
ZIP390	ACTION : ZIP	**m_Pkzip**	supported
	ACTION : UNZIP	**m_Pkunzip**	supported
	ENCRYPT		N.S.
	IFILE		supported
	OFILE		supported
	ARCHIVE		supported
	OVERWRITE		supported

Sort Utilities

Sort Utilities Equivalence Tables

SORT, SORTD, DFSORT, ICEMAN, SYNCSORTT

Table 2-5 Sort Utilities Equivalences

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

ICETOOL

Table 2-6 ICETOOL Equivalences

Sort Statement	Equivalent in Target Environment	MFSORT	SyncSort
SORT	m_FileSort	supported	supported
COPY	m_FileSort	supported	supported
MERGE	m_FileSort	supported	supported

DB2 Utilities

Only DB2 connect 9.7 or above is supported.

Table 2-7 DB2 Utilities

Command	Parameter	Equivalent in Target Environment	Oracle	DB2LUW
DSNTEP2	SQL	**m_ExecSQL**	supported	supported
DSNTEP4	SQL	**m_ExecSQL**	supported	supported
DSNTIAD	SQL	**m_ExecSQL**	supported	supported
DSNTIAUL	SQL	**m_ExecSQL**	supported	supported
	"Unload"	**m_DBTableUnload**	supported	supported
DSNUTILB	EXEC SQL	**m_ExecSQL**	supported	supported
	LOAD	**m_DBTableLoad**	supported	supported
	UNLOAD	**m_DBTableUnload**	supported	supported

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 [-c] [-d regexp] [-e] [-H] [-i] [-j] [-k] [-l] [-L] [-r] [-s] [-t file]
[-v] [-V n] Job
```

Arguments

-c
: Conversion phase (only available with TuxJes).

-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.

-e
: Execution phase (only available with TuxJes).

-H
: Execution stack (use limited to maintenance team).

-i
: Input phase (only available with TuxJes).

-j
: Job identifier (only available with TuxJes).

-k
: Cancel command (only available with TuxJes).

-l
: KSH listing included in log (only available with TuxJes).

-L
: Log directory (without TuxJes).

-r
: JES2 root (only available with TuxJes).

-s
: Sysout directory (without TuxJes).

-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 (use limited to maintenance team).

-v

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

-V n

Level mode (0 to 9).

Job

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

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 = Cnnnn, return code OK. • if MT_RC = Unnnn, User abort • if MT_RC= Syyy, System abort
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	Value fixed to D0000. The MT_RC return code is compared to this value to determine if the result of the execution was normal or not.
MT_RC_PROGRAM_ABORT	By default 127. Contains the return code from an executable (not applicative COBOL program). • from 0 to 127: return code OK. • from 128 to 142: it is a signal SKIL in MT_RC • more than 143: Unnn in MT_RC
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`) using the `-b` option, the runtime command, `runb`, connects it to the database according to the `MT_DB_LOGIN` environment variable . It must have the correct value (user name, password and Oracle instance, at least "/"). It can be used in the TuxJES Security Configuration file and specified when submitting jobs or set as an environment variable. If the configuration file is not specified, the environment variable `MT_DB_LOGIN` value is used.
- 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 the 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_FileLoad	m_OutputSet
m_CondEnd	m_FileOverride	m_PhaseBegin
m_CondExec	m_FilePrint	m_PhaseEnd
m_CondIf	m_FileRepro	m_ProcBegin
m_DBTableLoad	m_FileSort	m_ProcEnd
m_DirCreate	m_FileRename	m_ProcInclude
m_DirDelete	m_GenCommit	m_ProgramExec
m_DirRename	m_GenDefine	m_RcSet
m_ExecSQL	m_GenRollback	m_ShellInclude
m_FileAssign	m_JclLibSet	m_StepLibSet
m_FileBuild	m_JobBegin	m_SymbolDefault
m_FileClrData	m_JobEnd	m_SymbolSet
m_FileDelete	m_JobLibSet	m_UtilityExec
m_FileEmpty	m_OutputAssign	
m_FileExist	m_OutputOverride	

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. When <step> refers to a previous step, replace it with the step label, e.g. "`STEPEC01`". When <step> refers to a return code of the step in the procedure, replace it with "`STEP_PROCNAME_NUM`", where "`PROCNAME`" indicates the name of procedure, and "`NUM`" indicates the sequence number of the procedure calls.

Examples

```
m_CondExec EVEN
```

To refer to a step:

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

To refer to a return code of the step in procedure:

```
m_CondExec 4,LT,STEP_PROCNAME_NUM 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_DBTableLoad

Name

m_DBTableLoad – Loads the content of an input file into a database table.

Synopsis

```
m_DBTableLoad -t -i [-e] [-d] [-D]
```

Description

This function executes a command stored in the file which ddname is MT_CTL.

This command is either an SQLLDR command for Oracle or a DB2 LOAD command for DB2LUW according to the target database.

Options

-t <table name>
Mandatory option.
The name of the database table to be loaded.

-i <input flat file>
Mandatory option.
The ddname of the file containing the data to be loaded.

Options for Oracle

-e <error file>
Optional.

SYSERR by default.
The ddname of the error file where are stored the errors during sqlldr command execution.

-d <discard file>
Optional.

SYSERR by default.
The ddname of the file that contains the discarded data suring sqlldr command execution.

-D <discard file>
Optional.

999 by default

The maximum number of discarded data.

Options for DB2LUW

-e <error file>

Optional.

<Logfile> by default.
The ddname of the log file used during db2 load command execution.

-d <discard file>

Ignored.

-D <discard file>

Ignored.

Examples

```
       m_FileAssign -d OLD DDIN ${DATA}/MYINPUTFILE
       m_FileAssign -d SHR MT_CTL ${MT_CTL_FILES}/MYLOADCTL
       m_FileAssign -d SHR SYSERR ${[DATA}/MYSYSERR
m_DBTableLoad -t MYTABLE -i DDIN
```

m_DBTableUnload

Name

m_DBTableUnload – Unloads the content of a database table into a flat file.

Synopsis

```
m_DBTableUnload -t -o
```

Description

This function executes a command stored in the file which ddname is MT_CTL.

This command is either an SQLPLUS command for Oracle or a DB2 EXPORT command for DB2LUW according to the target database.

Options

-t <table name>
Mandatory option.
The name of the database table to be unloaded.

-o <output flat file>
Mandatory option.
The ddname of the file containing the unloaded data.

Examples

```
m_FileAssign -d OLD DDOUT ${DATA}/MYOUTPUTFILE

m_FileAssign -d SHR MT_CTL ${MT_CTL_FILES}/MYUNLOADCTL

m_DBTableUnload -t MYTABLE -o DDOUT
```

m_DirCopy

Name

m_DirCopy – Copies the members of a directory.

Synopsis

```
m_DirCopy [-i] [-o] [-s] [-e]
```

Description

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

Options

-i [(]{input|(input,R)}[:...][)]

Optional.

Default value : SYSUT1.

input: ddname of the input directory to be copied.

R: specifies that all members to be copied are to replace any identically named members in the output directory. When this option is specified, the ddname and R parameter must ce enclosed in a set of parentheses.

Several directories may appear separated by “:”

-o <output dir>

Optional.

Default value : SYSUT2.
The ddname of the output directory.

-s <member list>

Optional.

Member list to be copied (separated by comma and enclosed in a set of parentheses).

-e <member list>

Optional.

Member list to be excluded (separated by comma ans enclosed in a set of parentheses).

Examples

Copy of all the members from directory PDS1 to directory SEQ1.

```
m_DirCopy -i PDS1 -o SEQ1
```

Copy of the members A and K from directory PDS1 to directory SEQ1.

```
m_DirCopy -i PDS1 -s "(A,K)" -o SEQ1
```

Copy of all the members except member A from directory PDS1 to directory SEQ1.

```
m_DirCopy -i PDS1 -e A -o SEQ1
```

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 OldDirName NewDirName
```

Description

This function renames a directory.

Options

OldDirName
: The name of the directory to be renamed.

NewDirName
: The new name of the directory to be renamed.

m_ExecSQL

Name

m_ExecSQL — Executes an SQL script.

Synopsis

```
m_ExecSQL [-f] [-o]
```

Description

This function executes an SQL script.

The SQL directives (CREATE TABLE, CREATE INDEX, DELETE, SELECT …) are in the <inputfile> file. The results will be stored in the file <outputfile>.

Options

-f <inputfile>
The <inputfile> will contain the SQL directives (ddname of the file).

SYSIN is the default value.

-o <outputfile>
The <outputfile> will contain the results (ddname of the file).
SYSPRINT is the default value.

Examples

In the first example, the SQL directives are in the in-stream SYSIN file, the results will be stored in the SYSREC00 file.

```
       m_FileAssign -d ,CATLG SYSREC00 ${DATA}/FBACKE.LST.CUMUL

       m_FileAssign -i SYSIN

SELECT * FROM PJ01DB2.TABTEST2;

_end

       m_ExecSQL -o SYSREC00
```

In the second example, the SQL directives are the file TOW132C.sysin and the results will be printed.

```
m_OutputAssign -c “*” SYSPRINT

m_FileAssign -d SHR SYSIN ${SYSIN}/SYSIN/TOW132C.sysin

m_ExecSQL
```

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 -i [-C] [-D Delimiter] -g [CUR|ALL [-+np][rang]]
```

```
m_FileAssign -d [-C] [-r RecSize -t Type [-k Key]] -g [CUR|ALL [-+np][rang]]
ddname [-S 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

At least one of the two options “-i” and “-d” must be specified. All other options are optional.

`-C <concatenation>`

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

`-D <delimiter>`

Delimiter of sysin.

`-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]		
	[ANY]		

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.

An additional status has been added to the traditional z/OS status:
ANY — indicates to use a file in a special mode. The other sub-parameters (<normal-termination-disp> and <abnormal-termination-disp>) are ignored in this case.

Note: OLD and SHR check the file is already existing.

Note: NEW, creation if the file does not exist, abort if the file already exists.

MOD, creation if the file does not exist, continue if the file already exists.

<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 managed as KEEP option).

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

(CATLG is managed as KEEP option).
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 <argument>`

Indicates that the data set is a generation file. The possible values are:

+n: creates the nth new generation file.

-n: accesses the nth previous old generation file.

0: accesses the current generation.

CUR: accesses the current generation.

ALL: concatenates all generation.

`-i`

Indicates that the data set is a sysin.

`-k <key position>+>key length>`

Indicates the key characteristics of an indexed file (to be used when file organization is IDX).

`-r [<record length-min>-]<record length-max>`

Indicates the length characteristics of the record.

For fixed files, only the maximum record length is specified.

For variable files, the minimum and the maximum record lengths are specified.

`-S <model file>`

(S upper case character)
Names the dsname of a file (without extension ${DATA}).

When the disposition mode is NEW, this file is a “model” for the new file to be created.

The characteristics of the new file depends on the characteristics of the “model” file and other parameter given by the m_FileAssign function.

`-t <file organization>`

Indicates the file oraganization type:

SEQ: sequential

LSEQ: line sequential

REL: relative

IDX: indexed

PDS: directory

`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

```
Example with a shared file:

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

Example with a sysin and a delimiter:

       m_FileAssign -i -D FF INPUT

data input 1

data input 2

FF

Example with a sysin and continuation:

       m_FileAssign -i SYSIN

data input 1

_end

#%OPC BEGIN ACTION=INCLUDE

       m_FileAssign -i -C
```

```
data input 2
_end
#%OPC END ACTION=INCLUDE
       m_FileAssign -i -C
data input 3
_end
```

Example with a new file:

```
       m_FileAssign -d NEW -r 188 -t SEQ ENTREE ${DATA}/PJ01DDD.BT.KBIEI001
```

The new file will be a sequential file with a fixed record length of 188 bytes.

Example with a “model” file:

```
       m_FileAssign -d NEW -r 188 -S PJO1.MODEL ENTREE ${DATA}/PJ01.OUTPUT
```

The new file have the characteristics of the file PJ01.MODEL except for the record length given by the “-r” parameter.

m_FileBuild

Name

m_FileBuild — Creates a file.

Synopsis

```
m_FileBuild [-t][-r][-k][-K] [-S]<filename> dsname
```

Description

This function creates a file.

Options

-t <organization>

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

SEQ
: for a sequential file.

LSEQ
: for a line sequential file.

IDX
: for an indexed file

REL
: for a relative 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]

(K upper case character)

The secondary key indicating that the file contains a secondary key.

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

`Length`
: The length of the secondary key.

`d`
: Optional parameter.
Permit duplicates of secondary key.

-S <model file>

(S upper case character)
Names the dsname of a file (without extension ${DATA}).

When the disposition mode is NEW, this file is a “model” for the new file to be created.

The characteristics of the new file depends on the characteristics of the “model” file and other parameter given by the m_FileAssign function.

dsname

dsname of the file to create.

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}/METAW.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 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.

Example

```
m_FileEmpty -r MY_VARIABLE ${DATA}/rextest2

if [[ ${MY_VARIABLE} = true ]]; then

 echo "file is empty"

else

 echo "file is not empty"

fi
```

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.

Example

```
m_fileExist -r MY_VARIABLE ${DATA}/rextest2

if [[ ${MY_VARIABLE} = true ]]; then

 echo "file exists"

else

 echo "file does not exist"

fi
```

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 -i [-D Delimiter] -g [CUR|ALL [-+np][rang]] -s label ddname
```

```
m_FileOverride -d [-r RecSize -t Type [-k key]] -g [CUR|ALL [-+np][rang]]
-s label ddname dsname
```

Description

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

Options

Only "-s" and one of the two options "-i" and "-d" are mandatory to specify the usage. All other options are optional. See m_FileAssign for other options.

-s <label>
Name of the label in the called procedure.

Example

```
m_FileOverride -i -s PR3STEP1 SYSIN


m_FileOverride -d OLD -s MYSORT CUSTOM

${DATA}/BEAUSR2.QSAM.CUSTOM
```

m_FilePrint

Name

m_FilePrint — Prints a file (IDCAMS command PRINT).

Synopsis

```
m_FilePrint {-i ddname|-I dsname} {-o ddname|-O dsname}[-t][-C] [-S]
```

Description

This function prints a file.

Options

-C

Optional: Number of records to be listed.

-S

Optional: Number of records to skip before the listing begins.

-i infile|-I indataset

The input file is either a ddname (infile) or a dsname (indataset).

-o ddname|-O dsname

Optional: The output file is either a ddname (outfile) or a dsname (outdataset) (default value: SYSPRINT)

-t {CHAR|DUMP|HEX}

Optional: Type of print (default value: DUMP).

CHAR: specifies each byte of a record is to be listed as character.

DUMP: specifies each byte of a record is to be printed in both hexadecimal and character format.

HEX: specifies each byte of a record is to be listed as hexadecimal digits.

Example

```
m_FilePrint -I ${DATA}/INPUT -C 1

m_FilePrint -i INPUT -t CHAR -C 5
```

m_FileRepro

Name

m_FileRepro — Copies a file (IDCAMS command REPRO).

Synopsis

```
m_FileRepro {-i ddname|-I dsname} {-o ddname|-O dsname}[-C] [-S]
```

Description

This function copies a file.

Options

-C
Optional: Number of records to be copied.

-S
Optional: Number of records to skip before the copy begins.

-i infile|-I indataset
The input file is either a ddname (infile) or a dsname (indataset).

-o ddname|-O dsname
Optional: The output file is either a ddname (outfile) or a dsname (outdataset) (default value: SYSPRINT)

Example

```
m_FileRepro -I ${DATA}/INPUT -C 1

m_FileRepro -i INPUT -C 5
```

m_FileSort

Name

m_FileSort — Sorts a file.

Synopsis

```
m_FileSort -s SortSpecificationFile -i Infile [Infile ...] -o [Outfile]
```

Description

This function sorts a file.

Options

-s SortSpecification File
: The sort specification indicates either a file containing the sort specification or a file that indicates where the sort specification is to be found (ddname of file, by default SYSIN).

-i Infile
: At least one file must be used as input to the sort (ddname of file, by default SORTIN).

-o Outfile
: File to be used as output to the sort (ddname of file, by default SORTOUT).

Example with SyncSort commands

```
        m_FileAssign -i TOOLIN

        /FIELDS FLD1 5 CH 5

        /COND ...

        /OMIT ...

_end

        m_FileSort -s TOOLIN -i SORTIN -o SORTOUT
```

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_Ftp

Name

m_Ftp — Executes an FTP process.

Synopsis

```
m_Ftp -i <inputfile> [-e <ExitReturnCode>] [-n NETRC]
```

Description

This function reproduce the z/OS feature :EXEC PGM=FTP.

This function launches an ftp process and executes ftp command(s) stored in an input file.

An environment variable MT_FTP_TEST must be declared and initialized with the following value (upper or lower case):

Y or YES : test mode. The ftp commands different from “open”, “user”, “quit” or “bye” will not be executed.

N or NO: real mode. All the ftp commands will be executed.

Options

-i <inputfile>

Mandatory parameter.

ddname of the file which contains the ftp commands.

-e <ExitReturnCode>

Optional parameter.

Numeric return code to be returned when an error occured during ftp execution.

If ExitReturnCode = nn, the return code will be “C00nn” in case of ftp error.

If the parameter does not exist, the return code is always C0000.

-n NETRC

Optional parameter.

Necessary when the ftp connection is executed through “.netrc” file.

So “user” and “password” are omitted in <inputfile>.

Examples

```
       m_FileAssign -d ANY MT_LOC01 ${DATA}/ftp_file_loc1
       m_FileAssign -i SYSIN
open host
user user1 pw1
put MT_LOC01 DIR/file2
quit
_end
       m_Ftp -i INPUT
```

Notes

Some notes concerning the lines in the “SYSIN” file.

- The first line must be “open”:

format: open <HOSTNAME or ADDRESS>

example: open 172.20.12.21

- The second line must set the user ID and the password if the “-n NETRC” parameter is not set.

format: user <user> <password>

When the “-n NETRC” parameter is present, it implies that a “.netrc” file exist. This file identifies the user ID and the password to be used.

- The following lines are ftp’s commands (one per line) as get, put, ...

- The last line must be “quit”:

format: quit

m_GenCommit

Name

m_GenCommit — Commits a generation file.

Synopsis

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

Description

`m_GenCommit` commits a generation file. During the job execution, all created generations stay temporarily until the end of the job in which these generations are committed. This function permits an explicit commit of temporary generations before the job ends.

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 the generation file and generates a file ".gens" which contains the number of generations that should be kept (specified by option "-s").

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. During the job execution, all created generations stay temporarily until the end of the job in which these generations are committed. This function permits an explicit rollback of temporary generations before the job ends.

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 begin a job.

Synopsis

`m_JobBegin -j` *`jobname`* `[-C cond] [-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` *`condition`*
: Specifies the return code tests used to determine whether a job will continue processing or be terminated.

`-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][-w][-n][-d][-f][-H][-o][-D] ddname
```

Options

-c <class>
Class of the output queue.

-w <writer>

- INTRDR:

At the end of the m_ProgramExec function, submits the contents to TuxJes (hopefully a script shell).

- SMTP:

At the end of the m_ProgramExec function, submits the contents (hopefully SMTP commands) to send an email using the SMTP protocol.

-<writer>:

At the end of the m_ProgramExec function, submits the command associated with the <writer> name in the writer.conf file.

-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

Examples

Example with an output class A:

```
       m_OutputAssign -c A SYSOUT
```

Example with INTRDR:

In this case, the file which ddnmae is RDRCICO must contain a ksh script.

```
       m_OutputAssign -c R -w INTRDR RDRCICO
```

Example with an “output” reference:

The output EDI is referenced.

```
       m_OutputAssign -c A -o “*.EDI” SYSPRINT
```

m_OutputOverride

Name

`m_OutputOverride` – Overrides an output file.

Synopsis

```
m_OutputOverride [-c class][-n copies][-d dest][-f forms][-H][-o list of
output][-D dsname] -S Labelproc ddname
```

Description

This function overrides a sysout assignment (see m_OutputAssign).

Options

Only “-S” is mandatory, which specifies the step where the assignment is overridden. All other options are optional. See m_OutputAssign for other options.

-S <label>

Name of the label in the called procedure.

m_OutputSet

Name

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

It defines a reference and specifies associated processing options for sysout management.

Synopsis

```
m_OutputSet [-c][-n][-d][-f][-p][-w][-D] 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.

-w <external writer>
Specifies the use of an “external writer” to process the sysout file rather than TuxJes.

-D Y/N
Default reference (Y/N).

Y indicates that the reference can be implicitely referenced in following m_OutputAssign function calls.

N indicates that the reference can not be implicitely referenced in following m_OutputAssign function calls.

Reference
Reference name of the output. This name (and its associated characteristics) can be referenced in following m_OutputAssign function calls.

Example

```
 m_OutputSet -d LOCAL -D N EDI
m_OutputAssign -c L -o “*.EDI” SYSUT2
```

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_Pkzip

Name

m_Pkzip — Executes zip process.

Synopsis

```
m_Pkzip [-f][-F] -a [-k][-n][-g][-d]
```

Description

This function zips files, directories or members of directory in an archive.

Messages are reported in SYSPRINT file.

Note: the options “-f” and “-F” are not mandatory but at least one must be present.

Options

-f ‘<ddname1>[,<ddname2>[,<ddname3>]...]’

First format of “-f” option, optional.

ddname of the files to be zipped (enclosed with a single quote).

Multiples ddname must be separated by a comma (,).

-f ‘<dirname>[;<member1>...]’

Second format of “-f” option, optional.

dirname followed by member names to be zipped (enclosed with a single quote).

Member names must be separated by a semicolon (;).

When dirname is alone, all the members are zipped.

Wildcard (*or %) may be used (for member names only).

-F ‘<dsname1>[,<dsname2>[,<dsname3>]...]’

Optional.

dsname of the files to be zipped (enclosed with a single quote).

Multiples dsname must be separated by a comma (,).

-a <archive>

mandatory.

ddname of the archive in which zipped file(s) are stored.

-k <action>

Optional.

Action to be done (ADD by default).

Use UPDATE to overwrite in archive.

-n '<zipped_name>'

Optional.

Names of the zipped file(s) in the archive (enclosed with single quotes) separated by a comma (,).

When several files are zipped, the same order than done in option "-f" followed by option"-F" must be respected.

-g <Y|N>

Optional.

GZIP compatible format.

Reserved for future used. This option is not taken into account by the RunTime Batch.

-d '<zipdir>'

Optional.

Directory (enclosed by single quotes) where files(s) to zip are located (${DATA} by default).

Examples

In the following example, 2 files are to be zipped (ddname's FICIN01 and FIC02) named by the "-f" option, DDARCH (ddname of the archive) is named by the "-a" option while the first file will be renamed (see the "-n" option).

```
m_FileAssign -d SHR FICIN01 ${DATA}/PKZIP_FICIN01

m_FileAssign -d SHR FIC02 ${DATA}/FIC.FILE01

m_FileAssign -d NEW,CATLG DDARCH ${DATA}/ARCH.FILE

m_Pkzip -f 'FICIN01,FIC02' -a DDARCH -n 'NEW_FICIN01,'
```

The following example shows how to use at the same time the options “-f” “-F” and “-n”.

```
-f ‘ddn1,ddn2,ddn3,pds;member1;member2’

-F “dsn1,dsn2,dsn3’

-n ‘,newddn2,newwddn3,newmb1,,newdsn1,newdsn2,’
```

The ddn1, member2 and dsn3 are not renamed (they are replaced by comma(,)).

m_Pkunzip

Name

m_Pkunzip — Executes unzip process.

Synopsis

```
m_Pkunzip [-f][-F] -a [-k][-n][-g][-d][-o][-w]
```

Description

This function unzips files, directories or members of directory in an archive.

Messages are reported in SYSPRINT file.

Note: the options “-f” and “-F” are not mandatory but at least one must be present.

Options

-f ‘<ddname1>[,<ddname2>[,<ddname3>]...]’

Optional.

ddname of the outfile(s) to be unzipped (enclosed with a single quote).

Multiples ddname must be separated by a comma (,).

-F ‘<dsname1>[,<dsname2>[,<dsname3>]...]’

Optional.

dsname of the outfile(s) to be unzipped (enclosed with a single quote).

Multiples dsname must be separated by a comma (,).

-a <archive>

Mandatory.

ddname of the archive from which zipped file(s) are extracted.

-o <outddname>

Optional.

ddname of the outfile intowhich file(s) are to be extracted.

-k <action>

Optional.

Action to be done (EXTRACT by default).

Note: only EXTRACT is supported in this release.

-n '<zipped_name>'
Optional.

Names of the zipped file(s) in the archive (enclosed with single quotes) separated by a comma (,).

When several files are zipped, the same order than done in option "-f" followed by option"-F" must be respected.

-g <Y|N>
Optional.

GZUNZIP compatible format.

Reserved for future used. This option is not taken into account by the RunTime Batch.

-w <Y|N>
Optional.

Overwrite parameter (default N).Used to overwrite an existing file.

-d '<unzipdir>'
Optional.

Directory (enclosed by single quotes) where files(s) to unzip are to be stored (${DATA} by default).

Examples

Unzip all files of an archive without renaming files:

```
m_FileAssign -d NEW,CATLG DDARCHIV ${DATA}/PKZIP_ARCH

m_Pkunzip -a DDARCHIV
```

Unzip all files of an archive into an output file:

```
m_FileAssign -d NEW,CATLG DDARCHIV ${DATA}/PKZIP_ARCH

m_FileAssign -d NEW,CATLG OUTFILE ${DATA}/UNZIPPED.FILE

m_Pkunzip -a DDARCHIV -o OUTFILE
```

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 [param1=value1,param2=value2,…,paramN=valueN]
```

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:exit_name] Program [arguments]
```

Description

This function runs a COBOL program.

Options

`-b`

Indicates the database will be accessed by the program.

`-e exit_type:exit_name`

Indicates an exit routine should be used.

An exit routine may be used at the beginning and/or at the end of a program.

The exit type (BEGIN, END or BOTH) indicates if the exit routine must be called at the begin or at the end of the program (or both).

The exit name is used to build the names of the sub-programs to be inserted before the call to the program (RTEX-"exitName"-Begin) and after the call of the program (RTEX-"exitName"-End).

`Program [arguments]`

Program name and 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_ProgramExec -e BEGIN:EX1 BPRAB006
```

Indicates to run program BPRA006 after the call of the “exit routine” RTEX-EX1-Begin.

This “exit routine” written by the user may contain user actions (for example conerning “accounting”).

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_SendMail

Name

`m_SendMail` — Sends an email.

Synopsis

```
m_SendMail -t [-f] [-s] [-m] [-a] [-n] [-c] [-b]
```

Description

This function sends an e-mail.

When the option "-f" is omitted, the environment variable MT_FROM_ADDRESS_MAIL must be declared in the user's profile and initialized with the "From Adress" to be used by default.

Two environment variables are used (see RTBatch.conf file):

MT_SMTP_SERVER : SmtpServer (default value : "localhost")

MT_SMTP_PORT: SmtpPort (default value : "25")

Options

-t '<To-Address1>[,<To-Adress2>...]'

Mandatory option. At least, one "To-Address" must be declared.

Electronic mail address of the recipient(s) inclosed in single quotes.

Multiple addresses must be separated by a comma ",".

-f '<To-Address>'

Optional.

Electronic mail address of the sender inclosed in single quotes.

The environment variable MT_FROM_ADDRESS_MAIL is used when this option is missing.

-s "<Subject>"

Optional.

Subject of the email inclosed in double quotes.

-m <message file>

Optional.

ddname of the file containing the message of the email.

-a <attach-file1>[,<attach-file2>...]

Optional.

ddname of the attached document(s).

Multiple attachements must be separated by a comma ",".

-n <filename1>[,<filename2>...]

Optional.

Name of the attached document(s) referenced by option "-a".

Multiple names must be separated by a comma ",".

Note that there must have so many name as ddname.

-c <cc-Address1>[,<cc-Address2>...]

Optional.

Electronic mail address of the Copy Carbone recipient(s) inclosed in single quotes.

Multiple addresses must be separated by a comma ",".

-b <bcc-Address1>[,<bcc-Address2>...]

Optional.

Electronic mail address of the Blind Copy Carbone recipient(s) inclosed in single quotes.

Multiple addresses must be separated by a comma ",".

Example

In this example,

- the text of the message is stored in the sysin MESSAGE,

- the implicit "From-Address" is taken into the environment variable MT_FROM_ADDRESS_MAIL,

- the "To-Address" is "BOB" <BOB.FOSTER@USA.COM>,

- an attached document is to be sent (with ddname file CR1 and name FILE2.CSV).

```
m_FileAssign -i MESSAGE
text of the email
_end
m_FileAssign -d SHR CR1 ${DATA}/FPROD.FILE1
m_SendMail -t '"BOB" <BOB.FOSTER@USA.COM>' -m MESSAGE -a CR1 -n 'FILE1.CSV'
```

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_Smtp

Name

`m_Smtp` — Sends an email using SMTP protocol.

Synopsis

`m_Smtp -i`

Description

This function sends an e-mail using SMTP protocol.

The SMTP commands are stored in the input file.

The command uses the following format:

telnet <SmtpServer> <SmtpPort>

Two environment variables are used (see RTBatch.conf file):

MT_SMTP_SERVER : SmtpServer (default value : “localhost”)

MT_SMTP_PORT: SmtpPort (default value : “25”)

Options

-i <InputFile>
: Mandatory option.
ddname of the file containing the SMTP commands.

Example

```
m_Smtp -i SYSUT2
```

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
```

m_UtilityExec

Name

m_UtilityExec — Executes the stored commands.

Synopsis

`m_UtilityExec [sysin file]`

Description

Executes the contents of an external sysin. This function executes the script "UtilityName" in the current Shell. UtilityName is the physical name previously assigned to `[sysin file]`.

The script assigned with sysin should contain a sequence of RunTime Batch functions. Although this function can also launch native Korn shell scripts, it is recommended to use m_ProgramExec instead. In this way, assignation is not required.

Options

sysin file:
By default SYSIN.

ddname of a sysin file which contains function to launch.

Example

```
m_FileAssign -d NEW SYTSIN ${SYSIN}/SYSTIN/MUEX07.sysin

m_UtilityExec SYTSIN
```

The file MUEX07.sysin may have the following content:

m_ProgramExec -b ZVDL101

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 and the database connection (`MT_DB_LOGIN`). The output is a security profile file which contains the Oracle Tuxedo application password, user name, user password and the database connection.

Note: 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'`: Specifies the options passed to the EJR script file using the `-i` option.

`-i =scriptfile`: The script file.

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 l -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.

`changeconcurrent(chco) -g groupname -i serverid -n concurrent_num`

Changes the number of maximum concurrent executing jobs for the `ARTJESINITIATOR` server which is designated by the -g and -i options. The change takes effect with no need to restart the `ARTJESINITIATOR` server.
The options are as follows:

`-g groupname`: the Tuxedo group name of the `ARTJESINITIATOR` server

`-i serverid`: the Tuxedo server id of the `ARTJESINITIATOR` server

`-n concurrent_num`: the number of maximum concurrent executing jobs

The change is not persistent, which means the number is reset when the `ARTJESINITIATOR` server restarts.

printconcurrent(pco) -g groupname -i serverid

Displays the number of maximum concurrent executing jobs for the `ARTJESINITIATOR` server which is designated by -g and -i options. The options are as follows:

`-g groupname`: the Tuxedo group name of the `ARTJESINITIATOR` server

`-i serverid`: the Tuxedo server id of the `ARTJESINITIATOR` server

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.

`JOBREPOSITORY`
The path of the job repository where jobs are stored. The script file path inputted in job submitting may be a relative path in JOBREPOSITORY if it is set.

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 [-n
concurrent_num] [-d]”
```

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 jobclasses[jobclass]
Specifies the job classes this `ARTJESINITIATOR` server is associated. If this option is not specified, `ARTJESINITIATOR` associates with all job classes.

-n concurrent_num
Specifies the number of maximum concurrent executing jobs for this `ARTJESINITIATOR` server. The default value is 1.

-d
Specifies the number of maximum concurrent executing jobs for this `ARTJESINITIATOR` server can be change by `artjesadmin changeconcurrent` command.

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.