independIT Integrative Technologies GmbH Bergstraße 6 D–86529 Schrobenhausen



schedulix Server

Command Reference

Dieter Stubler

Ronald Jeninga

September 6, 2016

Copyright © 2016 independIT GmbH

Rechtlicher Hinweis

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung der independIT GmbH in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltsverzeichnis

In	halsverzeichnis	iii
Та	bellenverzeichnis	ix
I.	Allgemein	1
1.	Einleitung Einleitung	3 3
2.	Utilities Starten und Stoppen des Servers server-start server-stop sdmsh sdms-auto_restart sdms-get_variable sdms-rerun sdms-set_state sdms-set_variable sdms-set_warning sdms-submit	11 11 12 12 22 23 25 26 26 28 29
II.	User Commands	33
3.	alter commands alter comment alter environment alter event alter exit state mapping alter exit state profile alter folder alter footprint alter group alter interval alter job	35 36 38 39 40 41 42 44 45 46

	alter job definition	
	alter named resource	58
	alter resource	60
	alter resource state mapping	61
	alter resource state profile	62
	alter schedule	63
	alter scheduled event	65
	alter scope	66
	alter server	
	alter session	70
	alter trigger	72
	alter user	74
4.	connect commands	75
	connect	76
5	copy commands	79
J.	copy folder	80
	copy named resource	
	copy scope	
	copy scope	02
6.	create commands	83
	create comment	84
	create environment	86
	create event	87
	create exit state definition	88
	create exit state mapping	89
	create exit state profile	90
	create folder	93
	create footprint	95
	create group	97
	create interval	98
	create job definition	102
	create named resource	121
	create resource	124
	create resource state definition	128
	create resource state mapping	129
	create resource state profile	130
	create schedule	131
	create scheduled event	133
	create scope	135
	create trigger	138
	create user	

7.	deregister commands	149
	deregister	. 150
8.	disconnect commands	151
	disconnect	. 152
9.	drop commands	153
	drop comment	. 154
	drop environment	. 156
	drop event	. 157
	drop exit state definition	. 158
	drop exit state mapping	. 159
	drop exit state profile	. 160
	drop folder	. 161
	drop footprint	. 162
	drop group	
	drop interval	
	drop job definition	
	drop named resource	
	drop resource	
	drop resource state definition	
	drop resource state mapping	
	drop resource state profile	
	drop schedule	
	drop scheduled event	
	drop scope	
	drop trigger	
	drop user	
10	finish commands	177
	finish job	
	, wat a summanda	470
11	.get commands	179
	get parameter	
	get submittag	. 181
12	.kill commands	183
	kill session	. 184
13	link commands	185
	link resource	. 186
14	list commands	187
	list calendar	. 188

list dependency definition	189
list dependency hierarchy	
list environment	
list event	197
list exit state definition	198
list exit state mapping	199
list exit state profile	
list folder	
list footprint	
list group	
list interval	
list job	
list job definition hierarchy	
list named resource	
list resource state definition	
list resource state mapping	
list resource state profile	
list schedule	
list scheduled event	
list scope	
list session	
list trigger	
list user	
15. move commands	239
move folder	240
move job definition	241
move named resource	242
move schedule	243
move scope	244
16. multicommand commands	245
multicommand	246
17.register commands	247
register	
register	240
18. rename commands	249
rename environment	250
rename event	251
rename exit state definition	252
rename exit state mapping	253
rename exit state profile	
rename folder	

	rename footprint	
	rename group	
	rename interval	258
	rename job definition	259
	rename named resource	260
	rename resource state definition	261
	rename resource state mapping	262
	rename resource state profile	263
	rename schedule	264
	rename scope	265
	rename trigger	
	rename user	267
40		000
19	resume commands	269
	resume	270
20	select commands	271
	select	
		,
21	.set commands	273
	set parameter	274
22	show commands	275
	show comment	
	show environment	
	show event	
	show exit state definition	
	show exit state mapping	
	show exit state profile	
	show folder	
	show footprint	
	show group	
	show interval	
	show job	
	show job definition	
	show named resource	
	show resource	
	show resource state definition	
	show resource state mapping	
	show resource state profile	
	show schedule	
	show scheduled event	
	show scope	
	show session	352

	show system	354
	show trigger	356
	show user	360
23	shutdown commands	363
	shutdown	364
24	.stop commands	365
	stop server	366
25	submit commands	367
	submit	368
26	suspend commands	371
	suspend	372
III.	Jobserver Commands	373
27	.Jobserver Commands	375
	alter job	
	alter jobserver	
	connect	
	deregister	
	disconnect	
	get next job	
	multicommand	
	reassure	
	register	392
11.7	. Job Commands	202
		393
	.Job Commands	395
	alter job	396
	connect	
	disconnect	405
	get parameter	406
	get submittag	
	multicommand	
	set parameter	409
	set state	410
	submit	411

Tabellenverzeichnis

1.1.	Gültige Datumsformate
1.2.	Keywords die mit Quotes als Identifier verwendet werden dürfen
1.3.	Keywords und Synonyme
1.4.	Reservierte Worte
6.1.	job definition parameters
6.2.	Named Resource Parameter Typen
6.3.	Named Resource Usage
6.4.	job definition parameters
6.5.	List of triggertypes
11.1.	get parameter output
11.2.	get submittag output
14.1.	list dependency definition output
14.2.	list dependency hierarchy output
14.3.	list environment output
14.4.	list event output
14.5.	list exit state definition output
14.6.	list exit state mapping output
14.7.	list exit state profile output
14.8.	list folder output
14.9.	list footprint output
14.10.	list group output
14.11.	list interval output
14.12.	list job output
14.13.	list job definition hierarchy output
14.14.	list named resource output
14.15.	list resource state definition output
14.16.	list resource state mapping output
14.17.	list resource state profile output
14.18.	list schedule output
14.19.	list scheduled event output
14.20.	list scope output
14.21.	list session output
14.22.	list trigger output

14.23.	list user output	 . 237
22.1.	show comment output	 . 277
22.2.	show environment output	 . 279
22.3.	show environment RESOURCES Subtabelle	 . 279
22.4.	show environment JOB_DEFINITIONS Subtabelle	 . 280
22.5.	show event output	 . 282
22.6.	show event PARAMETERS Subtabelle	
22.7.	show exit state definition output	 . 283
22.8.	show exit state mapping output	
22.9.	show exit state mapping RANGES Subtabelle	
22.10.	show exit state profile output	
22.11.	show exit state profile STATES Subtabelle	 . 287
22.12.	show folder output	
22.13.	show folder PARAMETERS Subtabelle	
22.14.		
22.15.	show footprint output	
22.16.	show footprint RESOURCES Subtabelle	
22.17.	show footprint JOB_DEFINITIONS Subtabelle	
22.18.	show group output	 . 297
22.19.	show group MANAGE_PRIVS Subtabelle	
22.20.	show group USERS Subtabelle	 . 297
22.21.	show interval output	
22.22.	show interval SELECTION Subtabelle	
22.23.	show interval FILTER Subtabelle	 . 300
22.24.		
22.25.	show job CHILDREN Subtabelle	 . 308
22.26.	show job PARENTS Subtabelle	 . 309
22.27.	show job PARAMETER Subtabelle	
22.28.	show job REQUIRED_JOBS Subtabelle	 . 311
22.29.	show job DEPENDENT_JOBS Subtabelle	 . 314
22.30.	show job REQUIRED_RESOURCES Subtabelle	 . 315
22.31.	show job AUDIT_TRAIL Subtabelle	 . 316
22.32.	show job DEFINED_RESOURCES Subtabelle	 . 316
22.33.	show job definition output	 . 320
22.34.	show job definition CHILDREN Subtabelle	 . 321
22.35.	show job definition PARENTS Subtabelle	 . 322
22.36.	show job definition PARAMETER Subtabelle	 . 323
22.37.	show job definition REQUIRED_JOBS Subtabelle	 . 324
22.38.	show job definition DEPENDENT_JOBS Subtabelle	 . 325
22.39.	show job definition REQUIRED_RESOURCES Subtabelle	 . 326
22.40.	show named resource output	 . 328
22.41.	show named resource RESOURCES Subtabelle	 . 329

22.42.	show named resource PARAMETERS Subtabelle	329
22.43.	show named resource JOB_DEFINITIONS Subtabelle	330
22.44.	show resource output	333
22.45.	show resource ALLOCATIONS Subtabelle	334
22.46.	show resource PARAMETERS Subtabelle	335
22.47.	show resource state definition output	336
22.48.	show resource state mapping output	338
22.49.	show resource state mapping MAPPINGS Subtabelle	338
22.50.	show resource state profile output	
22.51.	show resource state profile STATES Subtabelle	340
22.52.	show schedule output	342
22.53.	show scheduled event output	
22.54.	show scope output	
22.55.	show scope RESOURCES Subtabelle	
22.56.	show scope CONFIG Subtabelle	350
22.57.	show scope CONFIG_ENVMAPPING Subtabelle	
22.58.	show scope PARAMETERS Subtabelle	
22.59.	show session output	353
22.60.	show system output	
22.61.	show system WORKER Subtabelle	355
22.62.	show trigger output	358
22.63.	show trigger STATES Subtabelle	359
22.64.	show user output	
22.65.	show user MANAGE_PRIVS Subtabelle	361
22.66.	show user GROUPS Subtabelle	361
22.67.	show user COMMENT Subtabelle	361
25.1.	submit output	370
27.1.	get next job output	389
28.1.	get parameter output	406
28.2.	get submittag output	
28.3.	submit output	413

Teil I. Allgemein

1. Einleitung

Einleitung

Dieses Dokument ist im Wesentlichen in drei Teilen gegliedert. Im BICsuite Scheduling System gibt es drei Arten von Benutzern (im weitesten Sinne des Wortes):

- Users
- Jobservers
- Jobs

Jeder dieser Benutzer hat einen eigenen Befehlssatz zur Verfügung. Diese Befehlssätze sind jeweils nur teilweise überlappend. Es gibt z.B. für Jobserver das **get next job** Statement, dass weder für Jobs noch für Users gültig ist. Dafür gibt es Formen des **submit** Statements die nur in einem Job-Kontext einen Sinn ergeben und daher auch nur von Jobs ausgeführt werden dürfen. Das Anlegen von Objekte wie Exit State Definitions oder Job Definitions ist natürlich nur den Usern vorbehalten. Im Gegensatz dazu gibt es auch Statements, wie z.B. das **connect** Statement, die für alle Arten von Benutzern gültig sind.

Die Gliederung dieses Dokumentes richtet sich nach den drei Arten von Benutzer. Der größte Teil des Dokumentes befasst sich mit den User Commands, die beide andere Teilen mit den Jobserver und Job Commands.

Zur Vollständigkeit wird im nächsten Kapitel noch kurz auf das Utility *sdmsh* eingegangen. Dieses Utility ist einfach zu handhaben und stellt eine excellente Wahl für die Verarbeitung von Skripten mit BICsuite Kommandos dar.

Da die im Folgenden beschriebenen Syntax die einzige Schnittstelle zum BICsuite Scheduling Server ist, benutzen alle Dienstprogramme, also insbesondere auch BICsuite! Web diese Schnittstelle.

Um die Entwicklung eigener Utilities zu vereinfachen kann der Server seine Reaktionen auf Statements in verschiedenen Formaten ausgeben. Das Utility *sdmsh* benutzt zum Beispiel das **serial** Protokoll in dem serialized Java Objects übermittelt werden. Dagegen benutzt BICsuite!Web das **python** Protokoll in dem textuelle Darstellungen von Python-Strukturen übermittelt werden, die mittels der eval()-Funktion leicht eingelesen werden können.

Syntax Diagramme

Die Syntax Diagramme sind aus verschiedenen Symbolen und Metasymbolen aufgebaut. Die Symbole und Metasymbole sind in der untenstehende Tabelle

Syntax Diagramme

T 3

Allgemein Einleitung

aufgeführt und erläutert.

Symbol	Bedeutung
keyword	Ein Schlüsselwort der Sprache. Diese müssen eingegeben werden, wie dargestellt. Ein Beispiel ist z.B. das Schlüsselwort create .
name	Ein Parameter. Hier kann in vielen Fällen ein selbst gewählter Name oder eine Zahl eingesetzt werden.
NONTERM	Ein nicht terminales Symbol wird mittels small caps dargestellt. An dieser Stelle muss ein, im späteren Verlauf des Diagrammes näher erläutertes, Syntaxele- ment eingesetzt werden.
< all any >	Dieses Syntaxelement stellt eine Wahlmöglichkeit dar. Einer der in den spitzen Klammern angegebene Syntaxelemente, das können natürlich auch nonterminale Symbole sein, muss gewählt werden. Im einfachsten Fall gibt es nur zwei Wahlmöglichkeiten, häufig aber auch mehr.
< <u>all</u> any >	Auch in diesem Fall handelt es sich um eine Wahlmöglichkeit. Im Gegensatz zum vorherigen Syntaxelement wird durch das Unterstreichen des ersten Elementes hervorgehoben, dass diese Wahlmöglichkeit der Default ist.
[or alter]	Optionale Syntaxelemente werden in eckigen Klammern gestellt.
{ statename }	Syntaxelemente die in geschweiften Klammern gestellt sind werden 0 bis n mal wiederholt.
JOB_PARAMETER	Der Fall, dass Elemente mindestens einmal vorkom-
{, JOB_PARAMETER}	men, ist weitaus häufiger und wird wie gezeigt dargestellt.
I	In Listen von möglichen Syntaxelementen, werden die einzelne Möglickeiten durch einen von einander ge- trennt. So eine Liste ist eine andere Darstellung ein- er Wahlmöglichkeit. Beide verschiedene Darstellungs- formen dienen jedoch der Übersichtlichkeit.

Literale

Literale

In der Sprachdefinition werden nur für Zeichenketten, Zahlen und Datum/Uhrzeit Angaben Literale benötigt.

Zeichenketten werden durch einfache Hochkommas abgegrenzt wie in

node = 'puma.independit.de'

Einleitung Allgemein

Ganze Zahlen werden entweder als vorzeichenlose *integer* oder vorzeichenbehaftete *signed_integer* in den Syntaxdiagramme dargestellt. Einem *signed_integer* darf einen Vorzeichen (+ oder -) vorangestellt werden. Gültige vorzeichenlose Integers liegen im Zahlenbereich zwischen 0 und $2^{31}-1$. Integers mit Vorzeichen liegen damit im Zahlenbereich zwischen $-2^{31}+1$ und $2^{31}-1$. Wenn in den Syntaxdiagramme die Rede von *id* ist, wird hier eine vorzeichenlose ganze Zahl zwischen 0 und $2^{63}-1$ erwartet.

Weitaus am schwierigsten sind die Datum/Uhrzeitangaben, insbesondere in den Statements die mit dem Time-Scheduling zu tun haben. Grundsätzlich werden diese Literale als Zeichenketten mit einem speziellen Format dargestellt.

Zur Vereinbarung der an ISO8601 angelehnten Notationen in Tabelle 1.1 wird folgende Schreibweise verwendet:

Zeichen	Bedeutung	zul.Bereich	Zeichen	Bedeutung	zul.Bereich
YYYY	Jahr	1970 9999	hh	Stunde	0023
MM	Monat	01 12	mm	Minute	00 59
DD	Tag (des Monats)	01 31	SS	Sekunde	00 59
ww	Woche (des Jahres)	01 53			

- Alle anderen Zeichen stehen für sich selbst.
- Es erfolgt keine Unterscheidung von Groß- und Kleinschreibung.
- Der früheste zulässige Zeitpunkt ist 1970–01–01T00:00:00 GMT.

Identifier

Innerhalb der BICsuite Scheduling System werden Objekte anhand ihres Namen identifiziert. (Strikt genommen können die Objekte auch über ihren internen Id, eine Nummer, identifiziert werden, aber diese Praxis wird nicht empfohlen). Gültige Namen bestehen aus einem Buchstabe, Underscore (_), At-Zeichen (@) oder Hash-Zeichen (#), gefolgt von Ziffern, Buchstaben oder genannte Sonderzeichen. Sprachabhängige Sonderzeichen wie z.B. deutsche Umlaute sind ungültig. Soweit Identifier nicht in einfachen Quotes eingeschlossen werden, werden sie ohne Berücksichtigung der Groß- oder Kleinschreibung behandelt. Werden sie jedoch in Quotes eingeschlossen, werden sie case-sensitive behandelt. Es wird daher generell nicht empfohlen Quotes zu benutzen, es sei denn es liegt ein triftiger Grund vor. Identifier die in einfachen Quotes eingeschlossen werden dürfen auch Leerzeichen

Identifier

enthalten. Auch hier gilt, dass diese Praxis nicht empfohlen wird, da Leerzeichen

Allgemein

Einleitung

Format	Beispiel	vereinfachtes Format
YYYY	1990	
YYYY-MM	1990-05	YYYYMM
YYYY-MM-DD	1990-05-02	YYYYMMDD
YYYY-MM-DDThh	1990-05-02T07	YYYYMMDDThh
YYYY-MM-DDThh:mm	1990-05-02T07:55	YYYYMMDDThhmm
YYYY-MM-DDThh:mm:ss	1990-05-02T07:55:12	YYYYMMDDThhmmss
-MM	-05	
-MM-DD	-05-02	-MMDD
-MM-DDThh	-05-02T07	-MMDDThh
-MM-DDThh:mm	-05 - 02T07:55	-MMDDThhmm
-MM-DDThh:mm:ss	-05-02T07:55:12	-MMDDThhmmss
DD	02	
DDThh	02T07	
DDThh:mm	02T07:55	DDThhmm
DDThh:mm:ss	02T07:55:12	DDThhmmss
Thh	T07	
Thh:mm	T07:55	Thhmm
Thh:mm:ss	T07:55:12	Thhmmss
T-mm	T-55	
T-mm:ss	T-55:12	T-mmss
Tss	T12	
YYYYWww	1990W18	
Www	W18	

Tabelle 1.1.: Gültige Datumsformate

normalerweise als Trennzeichen aufgefasst werden und daher leicht Fehler entstehen können. Insbesondere Leerzeichen am Ende des Namens führen leicht zu schwer auffindbaren Fehlern.

Es gibt eine Anzahl Schlüsselworte in der Syntax die nicht ohne Weiteres als Identifier benutzt werden können. Hier kann das Benutzen von Quotes sinnvoll sein, denn dadurch werden die Identifier nicht als Schlüsselworte erkannt. Die Tabelle 1.2 gibt eine Liste solcher Schlüsselworte.

Desweiteren gibt es eine Anzahl Synonymen. Das sind im Wesentlichen Schlüsselworte die mehrere Schreibweisen erlauben. In der Tabelle 1.2 wird nur einer diesen Schreibweisen gezeigt. Die Synonymen dürfen beliebig durcheinander benutzt werden. In der Tabelle 1.3 gibt es eine Liste solcher Synonymen.

Einleitung Allgemein

Wie in jeder Sprache gibt es auch reservierte Worte, bzw. Wortkombinationen. Eine Übersicht wird in der Tabelle 1.4 gezeigt. Bei den Wortpaare gilt als Besonderheit, dass ein Ersetzen des Leerzeichens durch ein Underscore ebenfalls ein reserviertes Wort ergibt. Das Wort named_resource ist damit auch reserviert. ("named#resource" jedoch nicht).

Versionen

Es gibt drei Versionen des BICsuite Scheduling System. Da Features der höheren Versionen nicht immer in den niedrigeren Versionen vorhanden sind, werden die dazu gehörige Statements, bzw. die entsprechenden Optionen innerhalb der Statements entsprechend gekennzeichnet. Oben an der äußeren Ecke der Seite wird mittels einer Buchstabe angegeben in welcher Version des Systems dieses Statement zur Verfügung steht. Abweichungen vom allgemeinen Statement werden im Syntaxdiagramm dargestellt.

Die Symbole haben folgende Bedeutung:

Symbol Bedeutung

- B Dieses Symbol kennzeichnet ein Feature der Basic und alle höhere Versionen.
- P Dieses Symbol kennzeichnet ein Feature der Professional und Enterprise Version.
- E Dieses Symbol kennzeichnet ein Feature der Enterprise Version.

Versionen

Allgemein Einleitung

activate	delay	group	milestone	rawpassword	submitcount
active	delete	header	minute	read	submittag
action	dependency	history	mode	reassure	submitted
add	deregister	hour	month	recursive	sum
after	dir	identified	move	register	suspend
alter	disable	ignore	multiplier	rename	sx
amount	disconnect	immediate	n	required	synchronizing
and	distribution	import	name	requestable	synctime
avg	drop	in	nicevalue	rerun	tag
base	dump	inactive	node	restartable	test
batch	duration	infinite	noinverse	restrict	time
before	dynamic	interval	nomaster	resume	timeout
broken	edit	inverse	nomerge	revoke	timestamp
by	embedded	is	nonfatal	rollback	to
cancel	enable	isx	nosuspend	run	touch
cancelled	endtime	ix	notrace	runnable	trace
cascade	environment	job	notrunc	running	translation
change	errlog	kill	nowarn	runtime	tree
check	error	killed	of	s	trigger
child	event	level	offline	sc	trunc
children	execute	liberal	on	schedule	type
childsuspend	expand	like	online	scope	update
childtag	expired	limits	only	selection	unreachable
clear	factor	line	or	serial	unresolved
command	failure	list	owner	server	usage
comment	fatal	local	parameters	session	use
condition	filter	lockmode	password	set	user
connect	final	logfile	path	shutdown	view
constant	finish	loops	pending	show	warn
content	finished	map	performance	sort	warning
copy	folder	maps	perl	started	week
count	footprint	mapping	pid	starting	with
create	for	master	pool	starttime	workdir
cycle	force	master_id	priority	static	x
day	free_amount	max	profile	status	xml
default	from	min	protocol	stop	year
definition	get	merge	public	strict	
defer	grant	merged	python	submit	

Tabelle 1.2.: Keywords die mit Quotes als Identifier verwendet werden dürfen

Einleitung Allgemein

Keyword	Synonym	Keyword	Synonym
definition	definitions	minute	minutes
dependency	dependencies	month	months
environment	environments	node	nodes
errlog	errlogfile	parameter	parameters
event	events	profile	profiles
folder	folders	resource	resources
footprint	footprints	schedule	schedules
grant	grants	scope	scopes
group	groups	server	servers
hour	hours	session	sessions
infinit	infinite	state	states, status
interval	intervals	translation	translations
job	jobs	user	users
mapping	mappings	week	weeks
milestone	milestones	year	years

Tabelle 1.3.: Keywords und Synonyme

after final	exit state translation	non fatal
all final	ext pid	requestable amount
backlog handling	finish child	resource state
before final	free amount	resource state definition
begin multicommand	get next job	resource state mapping
broken active	ignore dependency	resource state profile
broken finished	immediate local	resource template
change state	immediate merge	resource wait
default mapping	initial state	run program
dependency definition	job definition	rerun program
dependency hierarchy	job definition hierarchy	scheduled event
dependency mode	job final	state profile
dependency wait	job server	status mapping
end multicommand	job state	suspend limit
error text	keep final	submitting user
exec pid	kill program	synchronize wait
exit code	local constant	to kill
exit state	merge mode	until final
exit state mapping	merge global	until finished
exit state definition	merge local	
exit state profile	named resource	

Tabelle 1.4.: Reservierte Worte

2. Utilities

Starten und Stoppen des Servers

server-start

Einleitung

Das Utility server-start wird benutzt um den Scheduling Server zu starten.

Einleitung

Aufruf

Der Aufruf von server-start sieht folgendemaßen aus:

Aufruf

server-start [OPTIONS] *config-file*

OPTIONS:

-admin

-protected

Die einzelne Options haben folgende Bedeutung:

Option	Bedeutung
-admin	Der Server startet in "admin" Modus. Das be-
	deutet, dass User-Logins bis auf den Benutzer
	SYSTEM disabled sind.
-protected	Der Protected Mode ist vergleichbar mit
•	dem Admin Mode. Der Unterschied ist, dass
	auch die interne Threads (TimerThread und
	SchedulingThread) nicht gestartet werden.
	Damit können administrative Aufgaben
	erledigt werden, ohne dass nebenläufige
	Transaktionen durchgeführt werden.

Wenn der Server bereits gestartet ist, wird der zweite Server, je nach Konfiguration, entweder den Betrieb übernehmen, oder aber immer wieder erfolglos versuchen zu starten.

Utilities | 11

Allgemein sdmsh

Das Utility server-start kann nur vom Benutzer unter dessen Kennung das System installiert wurde, benutzt werden.

server-stop

Einleitung

Einleitung

Das Utility server-stop wird benutzt um den Scheduling Server zu stoppen.

Aufruf

Aufruf

Der Aufruf von server-stop sieht folgendemaßen aus:

server-stop

Es wird zuerst versucht den Server sanft an zu halten. Dabei werden zuerst alle User-Connections beendet um anschliessend alle internal Threads zu beenden.

Ist dieser Ansatz nicht erfolgreich, bzw. dauert es zu lange, wird der Server mit Betriebssystemmitteln beendet.

Wenn der Server nicht gestartet ist, hat die Benutzung des server-stop Befehls keine Auswirkung.

Das Utility server-stop kann nur vom Benutzer unter dessen Kennung das System installiert wurde, benutzt werden.

sdmsh

Einleitung

Einleitung

Das Utility *sdmsh* ist ein kleines Programm das ein interaktives Arbeiten mit dem Scheduling Server ermöglicht. Im Gegensatz zu etwa dem BICsuite!Web Frontend ist dieses Arbeiten textorientiert. Es ist damit möglich Skripte zu schreiben und diese über *sdmsh* ausführen zu lassen.

Das Executable *sdmsh* ist ein kleines Skript oder Batch-Datei, dass den Aufruf des benötigten Java-Programmes kapselt. Es spricht natürlich nichts dagegen, dieses Java-Programm mit der Hand auf zu rufen, das Skript dient nur dem Komfort.

Aufruf

Aufruf

Der Aufruf von sdmsh sieht folgendemaßen aus:

12 l sdmsh

В

sdmsh Allgemein

```
| < --port | -p > portnumber
| < --user | -u > username
| < --pass | -w > password
| < --jid | -j > jobid
| < --key | -k > jobkey
| < --[ no ]silent | -[ no ]s >
| < --[ no ]verbose | -[ no ]v >
| < --ini | -ini > inifile
| < --[ no ]tls | -[ no ]tls >
| --[ no ]help
| --info sessioninfo
| -[ no ]S
| --timeout timeout
```

Die einzelne Options haben folgende Bedeutung:

Option	Bedeutung
<host -h="" =""> hostname</host>	BICsuite!server Host
<port $ $ -p $>$ portnumber	BICsuite!server Port
<user -u="" =""> username</user>	Username (user oder jid muss specifiziert werden)
<pass -w="" =""> password</pass>	Password (wird in Komination mit der user Option verwendet)
<jid $ $ -j $>$ jobid	Job Id (user oder jid muss specifiziert werden)
<key $ $ -k $>$ $jobkey$	Job Key (wird in Kombination mit derjid Option verwendet)
<[no]silent -[no]s $>$	[Keine] (error) Meldungen werden nicht ausgegeben
<[no]verbose -[no]v $>$	[Keine] Kommandos, Feedbacks und zusätzliche Meldungen werden ausgegeben
<	Benutze die genannte Konfigurationsdatei zum Setzen von Optionen
$<$ [no]tls \mid -[no]tls $>$	Benutze den Zugang über TLS/SSL [nicht]
[no]help	Gebe einen Hilfetext aus
info sessioninfo	Setze die mitgegebene Information als beschreibende Information der Session

sdmsh I 13

Allgemein sdmsh

Option	Bedeutung
-[no]S	Silent Option. Die Option ist obsolet und aus
	Gründen der Rückwärtskompatibilität vorhan-
	den
timeout timeout	Die Anzahl Sekunden nach den der Server eine
	idle Session terminiert. Der Wert 0 bedeutet
	kein timeout

Selbstverständlich benötigt *sdmsh* Information um sich mit dem richtigen BICsuite Scheduling System zu verbinden. Dazu können die benötigten Daten auf der Kommandozeile oder mittels einer Options Datei spezifiziert werden. Fehlende Werte für Username und Password werden von *sdmsh* erfragt. Falls Werte für den Host und Port fehlen, werden die Defaults "localhost" und 2506 benutzt. Es wird nicht empfohlen das Password auf der Kommandozeile zu spezifizieren, da diese Information vielfach sehr einfach von anderen Benutzern gelesen werden kann.

Options Datei

Options Datei

Die Options Datei hat das Format einer Java Propertyfile. Für die genaue Syntaxspezifikation dieser verweisen wir auf die offizielle Java Dokumentation.

Es spielen folgende Optionsdateien eine Rolle:

- \$SDMSCONFIG/sdmshrc
- \$HOME/.sdmshrc
- Optional eine auf der Befehlszeile spezifizierte Datei

Die Dateien werden in der angegebene Reihenfolge ausgewertet. Falls Optionen in mehreren Dateien vorkommen "gewinnt" der Wert in der zuletzt ausgewertete Datei. Optionen die auf der Befehlszeile spezifiziert werden, haben Vorrang über allen anderen Spezifikationen.

Folgende Schlüsselworte werden erkannt:

sdmsh Allgemein

Keyword	Bedeutung
User	Name des Benutzers
Password	Password des Benutzers
Host	Name oder IP-Adresse des Hosts
Info	Zusätzliche Information zur Identifikation einer Connection wird gesetzt
Port	Portnumber des Scheduling Servers (Default: 2506)
Silent	(Fehler)Meldungen werden nicht ausgegeben
Timeout	Timeout Wert für die Session (0 ist kein Timeout)
TLS	Benutze eine SSL/TLS Connection
Verbose	Kommandos, Feedbacks und weitere Meldungen werden ausgegeben

Da das Password des Benutzers in Klartext in dieser Datei steht, muss sorgfaltig mit den Zugriffsrechte für diese Datei umgegangen werden. Es ist natürlich möglich das Password nicht zu spezifizieren und dieses bei jedem Start von *sdmsh* ein zu geben.

Die folgende Schlüsselworte sind ausschließlich in Konfigurationsdateien erlaubt:

Keyword	Bedeutung
KeyStore	Keystore für TLS/SSL Kommunication
TrustStore	Truststore für TLS/SSL Kommunication
KeyStorePassword	Keystore password
TrustStorePassword	Truststore password

Interne Befehle

Abgesehen von den in den nachfolgenden Kapiteln beschriebenen BICsuite Befehle, kennt *sdmsh* noch einige einfache eigene Befehle. Diese werden im Folgenden kurz beschrieben. Interne Befehle müssen nicht mit einem Semicolon abgeschlossen werden.

Interne Befehle

disconnect Mit dem disconnect Befehl wird *sdmsh* verlassen. Da in den verschiedenen Arbeitsumgebungen verschiedene Befehle für das Verlassen eines Tools gebräuchlich sind, wurde hier versucht auf viele Formulierungen zu hören. Die Syntax des disconnect Befehls ist:

```
< disconnect | bye | exit | quit >
```

BEISPIEL Hier folgt ein Beispiel für den disconnect Befehl.

sdmsh | 15

Allgemein sdmsh

```
ronald@jaguarundi:~$ sdmsh
Connect
CONNECT_TIME : 23 Aug 2007 07:13:30 GMT
Connected
[system@localhost:2506] SDMS> disconnect
ronald@jaguarundi:~$
```

echo Im Falle einer interaktiven Benutzung von *sdmsh* ist sichtbar welcher Befehl gerade eingegeben wurde. Dies ist im Batchbetrieb, d.h. beim Verarbeiten eines Skriptes nicht der Fall. Mit dem echo Befehl kann das Wiedergeben des eingegebenen Statements ein- und ausgeschaltet werden. Per Default ist es eingeschaltet. Die Syntax des echo Befehls ist:

```
echo < \underline{on} \mid off >
```

<u>BEISPIEL</u> Untenstehend wird der Effekt von beiden Möglichkeiten gezeigt. Nach dem Befehl **echo on**

```
[system@localhost:2506] SDMS> echo on
End of Output
[system@localhost:2506] SDMS> show session;
show session;
Session
    THIS : *
SESSIONID : 1001
    START : Tue Aug 23 11:47:34 GMT+01:00 2007
     USER : SYSTEM
     UID : 0
      IP : 127.0.0.1
     TXID: 136448
     IDLE : 0
  TIMEOUT : 0
STATEMENT : show session
Session shown
[system@localhost:2506] SDMS> echo off
End of Output
```

16 l sdmsh

sdmsh Allgemein

```
[system@localhost:2506] SDMS> show session;

Session

    THIS : *
SESSIONID : 1001
    START : Tue Aug 23 11:47:34 GMT+01:00 2007
    USER : SYSTEM
        UID : 0
        IP : 127.0.0.1
        TXID : 136457
        IDLE : 0
    TIMEOUT : 0
STATEMENT : show session

[system@localhost:2506] SDMS>
```

help Der help Befehl gibt eine kurze Hilfe zu den *sdmsh-*interne Befehle. Die Syntax des help Befehls ist:

help

<u>BEISPIEL</u> Der help Befehl gibt nur eine kurze Hilfe zur Syntax der internen *sdmsh*-Befehle aus. Dies ist ersichtlich im untenstehenden Beispiel. (Die Zeilen wurden für dieses Dokument umgebrochen, der tatsächliche Output kann daher vom Untenstehenden abweichen).

```
[system@localhost:2506] SDMS> help
Condensed Help Feature
Internal sdmsh Commands:
disconnect|bye|exit|quit
                                 -- leaves the tool
echo on|off
                                 -- controls whether the statementtext is
                                   printed or not
help
                                 -- gives this output
include '<filespec>'
                                 -- reads sdms(h) commands from the given
                                    file
prompt '<somestring>'
                                 -- sets to prompt to the specified value
                                    %H = hostname, %P = port, %U = user,
                                    응응 = 응
timing on | off
                                 -- controls whether the actual time is
                                    printed or not
whenever error
 continue|disconnect <integer> -- specifies the behaviour of the program
                                    in case of an error
```

sdmsh I 17

Allgemein sdmsh

include Mit dem include Befehl können Dateien mit BICsuite Statements eingebunden werden.

Die Syntax des include Befehls ist:

include 'filespec'

<u>BEISPIEL</u> Im folgenden Beispiel wird eine Datei die nur den Befehl "**show session**;" enthält eingefügt.

```
[system@localhost:2506] SDMS> include '/tmp/show.sdms'
Session

THIS : *
SESSIONID : 1001
    START : Tue Aug 23 11:47:34 GMT+01:00 2007
    USER : SYSTEM
    UID : 0
        IP : 127.0.0.1
    TXID : 136493
    IDLE : 0
TIMEOUT : 0
STATEMENT : show session

Session shown
[system@localhost:2506] SDMS>
```

prompt Mit dem prompt Befehl kann ein beliebiger Prompt spezifiziert werden. Dabei gibt es eine Anzahl von variable Werte die vom Programm automatisch eingefügt werden können.

In untenstehender Tabelle stehen die Codes für die einzelne Variablen:

sdmsh Allgemein

Code	Bedeutung
%H	Hostname des Scheduling Servers
% P	TCP/IP Port
%U	Username
%% ***********************************	Prozentzeichen (%)

Der Default Prompt hat folgende Definition: [%U@%H:%P] SDMS>. Die Syntax des prompt Befehls ist:

prompt 'somestring'

<u>BEISPIEL</u> Im folgenden Beispiel wird zuerst ein leerer Prompt definiert. Daraufhin wird, um den Effekt deutlicher Sichtbar zu machen, ein BICsuite Statement ausgeführt. Anschliessend wird eine einfache Zeichenkette als Prompt gewählt und zum Schluss werden die Variablen benutzt.

```
[system@localhost:2506] SDMS> prompt ''
End of Output
show session;
show session;
Session
    THIS: *
SESSIONID : 1001
   START : Tue Aug 23 11:47:34 GMT+01:00 2007
    USER : SYSTEM
     UID : 0
      IP : 127.0.0.1
    TXID : 136532
    IDLE : 0
  TIMEOUT : 0
STATEMENT : show session
Session shown
prompt 'hello world '
End of Output
hello world prompt '[%U0%H:%P] please enter your wish! > '
End of Output
[system@localhost:2506] please enter your wish! >
```

sdmsh I 19

Allgemein sdmsh

timing Mit dem timing Befehl bekommt man Information bezüglich der Ausführungszeit eines Statements. Normalerweise ist diese Option ausgeschaltet und wird also keine Angabe der Ausführungszeit gemacht. Die Angaben erfolgen in Millisekunden.

Die Syntax des timing Befehls ist:

```
timing < off | on >
```

<u>BEISPIEL</u> Im folgenden Beispiel wird timing Information für ein einfaches BICsuite Statement gezeigt. Sichtbar wird die Ausführungszeit des Statements sowie die Zeit die zum Ausgeben des Resultats benötigt war.

```
[system@localhost:2506] SDMS> timing on
End of Output
[system@localhost:2506] SDMS> show session;
Execution Time: 63
show session;
Session
    THIS : *
SESSIONID: 1002
   START: Tue Aug 23 11:53:15 GMT+01:00 2007
    USER : SYSTEM
     UID : 0
      IP : 127.0.0.1
     TXID: 136559
    IDLE : 0
  TIMEOUT : 0
STATEMENT : show session
Session shown
[system@localhost:2506] SDMS>
Render Time : 143
```

whenever Insbesondere wenn *sdmsh* zur Ausführung von Skripte benutzt wird, ist eine Möglichkeit zur Fehlerbehandlung unerlässlich. Mit dem whenever Statement wird *sdmsh* gesagt wie es mit Fehlern umgehen soll. Defaultmäßig werden Fehler ignoriert, was für ein interaktives Arbeiten auch dem gewünschten Verhalten entspricht.

Die Syntax des whenever Statements ist:

whenever error < continue | disconnect integer >

20 l sdmsh

Allgemein

В

sdmsh

<u>BEISPIEL</u> Das untenstehende Beispiel zeigt sowohl das Verhalten von der **continue**-Option als auch das Verhalten der **disconnect-**Option. Der Exit Code eines Prozesses die von der Bourne-Shell gestartet wurde (und auch andere Unix-Shells) kann durch ausgabe der Variable \$? sichtbar gemacht werden.

```
[system@localhost:2506] SDMS> whenever error continue
End of Output

[system@localhost:2506] SDMS> show exit state definition does_not_exist;
show exit state definition does_not_exist;

ERROR:03201292040, DOES_NOT_EXIST not found

[system@localhost:2506] SDMS> whenever error disconnect 17

End of Output

[system@localhost:2506] SDMS> show exit state definition does_not_exist;
show exit state definition does_not_exist;

ERROR:03201292040, DOES_NOT_EXIST not found

[system@localhost:2506] SDMS> ronald@jaguarundi:~$ echo $?
17
ronald@jaguarundi:~$
```

Shellaufruf Es kommt häufig vor, dass mal schnell ein Shell-Kommando abgesetzt werden muss, etwa um kurz zu sehen wie die Datei die man (mittels **include**) ausführen möchte auch wieder heisst. Soweit keine besondere Fähigkeiten vom Terminal verlangt werden, wie das z.B. bei einem Aufruf eines Editors der Fall wäre, kann ein Shell-Kommando durch das Voranstellen eines Ausrufezeichens ausgeführt werden.

Die Syntax eines Shellaufrufes ist:

!shellcommand

<u>BEISPIEL</u> Im folgenden Beispiel wird kurz eine Liste aller *sdmsh* Skripte im /tmp Verzeichnis ausgegeben.

```
[system@localhost:2506] SDMS> !ls -l /tmp/*.sdms
-rw-r--r-- 1 ronald ronald 15 2007-08-23 09:30 /tmp/ls.sdms
End of Output
[system@localhost:2506] SDMS>
```

sdmsh | 21

sdms-auto_restart

sdms-auto_restart

Einleitung

Einleitung

Das Utility sdms-auto_restart dient dazu Jobs die fehlgeschlagen sind automatisch zu restarten. Dazu müssen eine Anzahl einfache Voraussetzungen erfüllt sein. Die wohl wichtigste Voraussetzung ist, dass der Job einen Parameter AUTORESTART mit dem Wert TRUE definiert. Natürlich kann dieser Parameter auch auf höhere Ebene gesetzt sein.

Folgende Parameter haben einen Einfluß auf das Verhalten des Autorestart-Utilities:

Parameter	Wirkung
AUTORESTART	Der Autorestart funktioniert nur wenn
	dieser Parameter den Wert "TRUE" hat
AUTORESTART_MAX	Definiert, wenn gesetzt, die maximale An-
	zahl automatischen Restarts
AUTORESTART_COUNT	Wird vom aurorestart Utility gesetzt um die
	Anzahl Restarts zu zählen
AUTORESTART_DELAY	Die Zeit in Minuten bevor ein Job erneut ges-
	tartet wird

Das Autorestart Utility kann als Trigger definiert werden. Hierfür bieten sich die Triggertypen IMMEDIATE_LOCAL sowie FINISH_CHILD an.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmsauto_restart Anwendung.

Aufruf

Aufruf

Der Aufruf von sdms-auto_restart sieht folgendemaßen aus:

```
 \begin{aligned} & \textbf{sdms-auto\_restart} \; [\; \texttt{OPTIONS} \; ] < \textbf{--host} \; | \; \textbf{-h} > hostname \\ & < \textbf{--port} \; | \; \textbf{-p} > portnumber < \textbf{--user} \; | \; \textbf{-u} > username \\ & < \textbf{--pass} \; | \; \textbf{-w} > password < \textbf{--failed} \; | \; \textbf{-f} > jobid \end{aligned}
```

OPTIONS:

```
< --silent | -s >
< --verbose | -v >
< --timeout | -t > minutes
< --cycle | -c > minutes
< --help | -h >
< --delay | -d > seconds
< --max | -m > number
< --warn | -W >
```

Allgemein

Die einzelne Options haben folgende Bedeutung:

Option	Bedeutung
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers
< port $ $ -p $>$ $portnumber$	Port des Scheduling Servers
<user $ $ -u $>$ $username$	Username für die Anmeldung
<pass $ $ -w $>$ password	Password für die Anmeldung
<failed $ $ -f $>$ $jobid$	Jobid des zu restartenden Jobs
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen
<timeout $ $ -t $>$ $minutes$	Anzahl Minuten die versucht wird um einen
	Server-Connection zu bekommen
<cycle $ $ -c $>$ $minutes$	Anzahl Minuten die zwischen zwei Ver-
	suchen gewartet wird
$<$ help \mid -h $>$	Gibt eine kurze Hilfe aus
<delay $ $ -d $>$ $minutes$	Die Anzahl Minuten die gewartet wird, bis
	der Job erneut gestartet wird
<max $ $ -m $>$ $number$	Die maximum Anzahl automatische
	Restarts
<warn -w="" =""></warn>	Das Warning Flag wird gesetzt, wenn die maximum Anzahl Restarts erreicht wurde

sdms-get_variable

Einleitung

Das Utility *sdms-get_variable* bietet eine einfache Möglichkeit Job-Parameter aus Einleitung dem Scheduling System zu lesen.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmsget_variable Anwendung.

Aufruf

Der Aufruf von sdms-get_variable sieht folgendemaßen aus:

Aufruf

$$\label{eq:sdms-get_variable} $$ $$ $$ sdms-get_variable [OPTIONS] < --host | -h > hostname $$ < --port | -p > portnumber < --jid | -j > jobid $$ < --name | -n > parametername $$$$

Allgemein

sdms-get_variable

Die einzelne Options haben folgende Bedeutung:

Option	Bedeutung
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers
<port $ $ -p $>$ portnumber	Port des Scheduling Servers
<user $ $ -u $>$ $username$	Username für die Anmeldung
<pass -w="" =""> password</pass>	Password für die Anmeldung (für eine Connection als User)
<key $ $ -k $>$ $jobkey$	Password für die Anmeldung (für eine Connection als Job)
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen
<timeout -t="" =""> <i>minutes</i></timeout>	Anzahl Minuten die versucht wird um einen Server-Connection zu bekommen
<cycle -c="" =""> minutes</cycle>	Anzahl Minuten die zwischen zwei Versuchen eine Server-Connection auf zu bauen gewartet wird
<help $ $ -h $>$	Gibt eine kurze Hilfe zum Aufruf des Utilities aus
<mode -m="" =""> mode</mode>	Modus für die Parameterermittlung (liberal, warn, strict)

Beispiel

Beispiel

```
ronald@cheetah:~$ sdms-get_variable -h localhost -p 2506 \
-j 5175119 -u donald -w duck -n ANTWORT
42
```

sdms-rerun Allgemein

sdms-rerun

Einleitung

Das Utility *sdms-rerun* wird genutzt um aus einem Skript oder Programm heraus *Einleitung* einen Job in restartable State erneut zu starten.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmsrerun Anwendung.

Aufruf

Der Aufruf von sdms-rerun sieht folgendemaßen aus:

Aufruf

OPTIONS:

```
<--user | -u > username

<--pass | -w > password

<--key | -k > jobkey

<--silent | -s >

<--verbose | -v >

<--timeout | -t > minutes

<--cycle | -c > minutes

<--help | -h >

<--suspend | -S >

<--delay | -D > delay

<--unit | -U > unit

<--at | -A > at
```

Die einzelne Options haben folgende Bedeutung:

sdms-rerun I 25

Allgemein sdms-set_variable

Option	Bedeutung		
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers		
<port $ $ -p $>$ portnumber	Port des Scheduling Servers		
<user $ $ -u $>$ $username$	Username für die Anmeldung		
<pass $ $ -w $>$ password	Password für die Anmeldung (für eine Con-		
	nection als User)		
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen		
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen		
<timeout $ $ -t $>$ $minutes$	Anzahl Minuten die versucht wird um einen		
	Server-Connection zu bekommen		
<cycle $ $ -c $>$ $minutes$	Anzahl Minuten die zwischen zwei Ver-		
	suchen eine Server-Connection auf zu bauen		
	gewartet wird		
$<$ help \mid -h $>$	Gibt eine kurze Hilfe zum Aufruf des Utili-		
	ties aus		
$<$ suspend \mid -S $>$	Der Job wird suspended		
<delay $ $ -D $>$ delay	Nach delay Units wird der Job automatisch		
	resumed		
$<$ unit \mid -U $>$ $unit$	Einheit für die delay-Option (default		
	MINUTE)		
<at $ $ -A $>$ at	Automatischer Resume zum angegebenen		
	Zeitpunkt		

sdms-set_state

Einleitung

Einleitung

sdms-set_variable

Einleitung

Einleitung

Das Utility *sdms-set_variable* bietet eine einfache Möglichkeit Job-Parameter im Scheduling System zu setzen.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmsset_variable Anwendung.

Aufruf

Aufruf

Der Aufruf von sdms-set_variable sieht folgendemaßen aus:

```
sdms-set_variable [ OPTIONS ] < --host | -h > hostname < --port | -p > portnumber < --jid | -j > jobid parametername wert { parametername wert}
```

Die einzelne Options haben folgende Bedeutung:

Option	Bedeutung
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers
<port $ $ -p $>$ portnumber	Port des Scheduling Servers
<user $ $ -u $>$ $username$	Username für die Anmeldung
<pass -w="" =""> password</pass>	Password für die Anmeldung (für eine Connection als User)
<key $ $ -k $>$ $jobkey$	Password für die Anmeldung (für eine Connection als Job)
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen
<timeout -t="" =""> minutes</timeout>	Anzahl Minuten die versucht wird um einen
<cycle -c="" =""> minutes</cycle>	Server-Connection zu bekommen Anzahl Minuten die zwischen zwei Versuchen eine Server-Connection auf zu bauen gewartet wird
$<$ help \mid -h $>$	Gibt eine kurze Hilfe zum Aufruf des Utilities aus
<case -c="" =""></case>	Namen sind Case Sensitive

Allgemein

sdms-set_warning

sdms-set_warning

Einleitung

Einleitung

Das Utility *sdms-set_warning* wird benutzt um das Warning-Flag eines Jobs zu setzen. Optional kann ein Text spezifiziert werden. Man kann als Benutzer für einen Job das Warning Flag setzen wenn man das Operate-Privileg hat. Ein Job kann das Warning-Flag für sich selbst setzen.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmsset_warning Anwendung.

Aufruf

Aufruf Der Aufruf von sdms-set_warning sieht folgendemaßen aus:

```
sdms-set_warning [ OPTIONS ] < --host | -h > hostname < --port | -p > portnumber < --jid | -j > jobid
```

OPTIONS:

```
< --user | -u > username
< --pass | -w > password
< --key | -k > jobkey
< --silent | -s >
< --verbose | -v >
< --timeout | -t > minutes
< --cycle | -c > minutes
< --help | -h >
< --warning | -m > warning
```

Die einzelne Options haben folgende Bedeutung:

emein

Option	Bedeutung
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers
<port $ $ -p $>$ portnumber	Port des Scheduling Servers
<user $ $ -u $>$ $username$	Username für die Anmeldung
<pass $ $ -w $>$ $password$	Password für die Anmeldung (für eine Connection als User)
<key $ $ -k $>$ $jobkey$	Password für die Anmeldung (für eine Connection als Job)
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen
<timeout -t="" =""> <i>minutes</i></timeout>	Anzahl Minuten die versucht wird um einen Server-Connection zu bekommen
<cycle -c="" =""> minutes</cycle>	Anzahl Minuten die zwischen zwei Versuchen eine Server-Connection auf zu bauen gewartet wird
<help -h="" =""></help>	Gibt eine kurze Hilfe zum Aufruf des Utilities aus
<warning -m="" =""> warning</warning>	Text zur Warnung

sdms-submit

Einleitung

Das Utility *sdms-submit* wird benutzt um Jobs oder Batches zu starten. Diese können als eigenständiger Ablauf oder aber als Kind eines bereits vorhandenen Jobs gestartet werden. Im letzteren Fall kann, wenn in der Parent-Child Hierarchie definiert, ein Alias zur Identifizierung des zu submittenden Jobs oder Batches spezifiziert werden.

Die Logik der Optionsdateien die für das Utility sdmsh gilt, findet auch für sdmssubmit Anwendung.

Aufruf

Der Aufruf von sdms-submit sieht folgendemaßen aus:

Aufruf

Einleitung

```
sdms-submit [ OPTIONS ] < --host | -h > hostname
< --port | -p > portnumber < --job | -J > jobname

OPTIONS:
    < --user | -u > username
    < --pass | -w > password
    < --jid | -j > jobid
```

Allgemein

sdms-submit

Die einzelne Options haben folgende Bedeutung:

sdms-submit Allgemein

Option	Bedeutung			
<host -h="" =""> hostname</host>	Hostname des Scheduling Servers			
< port $ $ -p $>$ $portnumber$	Port des Scheduling Servers			
<user $ $ -u $>$ $username$	Username für die Anmeldung			
<pass -w="" =""> password</pass>	Password für die Anmeldung (für eine Connection als User)			
<key $ $ -k $>$ $jobkey$	Password für die Anmeldung (für eine Connection als Job)			
$<$ silent \mid -s $>$	Reduzierte Augabe von Meldungen			
$<$ verbose \mid -v $>$	Erhöhte Anzahl Meldungen			
<timeout $ $ -t $>$ $minutes$	Anzahl Minuten die versucht wird um einen			
	Server-Connection zu bekommen			
<cycle -c="" =""> minutes</cycle>	Anzahl Minuten die zwischen zwei Ver-			
	suchen eine Server-Connection auf zu bauen gewartet wird			
<help -h="" =""></help>	Gibt eine kurze Hilfe zum Aufruf des Utili-			
· itel	ties aus			
<tag $ $ -T $>$ tag	Tag für dynamic Submits			
<master -m="" =""></master>	Submit eines Masters, kein Child			
<suspend -s="" =""></suspend>	Der Job wird suspended			
<delay $ $ -D $>$ $delay$	Nach delay Units wird der Job automatisch			
	resumed			
<unit -u="" =""> <i>unit</i></unit>	Einheit für die delay-Option (default MINUTE)			
<at $ $ -A $>$ at	Automatischer Resume zum angegebenen Zeitpunkt			

Teil II. User Commands

Utilities I 33

3. alter commands

alter comment

alter comment

Zweck

Zweck

Das alter comment Statement wird eingesetzt um den Kommentar zu einem Objekt zu ändern.

Syntax

Syntax

Die Syntax des alter comment Statements ist

```
alter [ existing ] comment on OBJECTURL with CC_WITHITEM
```

OBJECTURL:

distribution distributionname for pool resourcepath in serverpath

environment environmentname

exit state definition statename

exit state mapping mappingname

exit state profile profilename

event eventname

resource resourcepath in folderpath

folder folderpath

footprint footprintname

group groupname

interval intervalname

job definition folderpath

job jobid

named resource resourcepath

parameter parametername of PARAM_LOC

resource state definition statename

resource state mapping mappingname

resource state profile profilename

scheduled event schedulepath . eventname

schedule schedulepath

resource resourcepath in serverpath

< **scope** *serverpath* | **job server** *serverpath* >

 $trigger \ triggername \ on \ {\tt TRIGGEROBJECT} \ [\ < noinverse \ | \ inverse >]$

user username

CC_WITHITEM:

CC_TEXTITEM {, CC_TEXTITEM}

alter comment

User Commands

```
PARAM_LOC:
    folder folderpath
    | job definition folderpath
    | < scope serverpath | job server serverpath >

TRIGGEROBJECT:
    resource resourcepath in folderpath
    | job definition folderpath
    | object monitor objecttypename
    | resource resourcepath in serverpath

CC_TEXTITEM:
    tag = < none | string > , text = string
    | text = string
```

Beschreibung

Der alter comment Befehl wird verwendet um die Kurzbeschreibung bwz. die URL der Beschreibung von dem beschriebenen Objekt zu ändern. Natürlich kann der Typ der Information ebenso verändert werden. Der Kommentar ist versioniert. Das bedeutet, dass Kommentare nicht überschrieben werden. Wenn das kommentierte Objekt angezeigt wird, ist der angezeigte Kommentar, der Kommentar der zur Version des angezeigten Objektes passt.

Der optionale Schlüsselwort **existing** wird verwendet um das Auftreten der Fehlermeldungen und das Abbrechen der aktuellen Durchführung zu verhindern. Das ist in Zusammenhang mit multicommands besonders nützlich.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Beschreibung

alter environment

alter environment

Zweck

Zweck

Das alter environment Statement wird eingesetzt um die Eigenschaften des spezifizierten Environments zu ändern.

Syntax

Syntax

Die Syntax des alter environment Statements ist

```
alter [ existing ] environment environmentname
with ENV_WITH_ITEM

alter [ existing ] environment environmentname
add ( ENV_RESOURCE {, ENV_RESOURCE} )

alter [ existing ] environment environmentname
delete ( resourcepath {, resourcepath} )

ENV_WITH_ITEM:
    resource = none
    | resource = ( ENV_RESOURCE {, ENV_RESOURCE} )

ENV_RESOURCE:
    resourcepath [ < condition = string | condition = none > ]
```

Beschreibung

Beschreibung

Das alter environment Statement wird benutzt um die Resource-Anforderungen, die in diesem Envrionment definiert sind, zu ändern. Laufende Jobs sind nicht davon betroffen.

Die "with resource =" Form des Statements ersetzt die aktuelle Gruppe von Resource-Anforderungen. Die anderen Arten fügen die spezifizierten Anforderungen zu oder löschen sie. Es wird als Fehler angesehen eine Anforderung zu löschen welche kein Teil des dem Environments ist oder eine Anforderung für eine bereits benötigte Resource zuzufügen.

Nur Administratoren sind befugt diese Handlung durchzuführen.

Ausgabe

Ausgabe

alter event

User Commands

alter event

Zweck

Das alter event Statement wird eingesetzt um Eigenschaften des spezifizierten Zweck Events zu ändern.

Syntax

Die Syntax des alter event Statements ist

Syntax

```
alter [ existing ] event eventname
with EVENT_WITHITEM {, EVENT_WITHITEM}

EVENT_WITHITEM:
    action =
    submit folderpath [ with parameter = ( PARAM {, PARAM} ) ]
    | group = groupname

PARAM:
    parametername = < string | number >
```

Beschreibung

Das alter event statement wird benutzt um die Eigenschaften eines Events zu ändern. Mit der **with parameter** Klausel kann ein Parameter für den Submit eines Jobs spezifiziert werden. Für eine ausführliche Beschreibung der Optionen, siehe das create event Statement auf Seite 87.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

alter exit state mapping

alter exit state mapping

Zweck

Zweck

Das alter exist state mapping Statement wird eingesetzt um Eigenschaften des spezifizierten Mappings zu ändern.

Syntax

Syntax

Die Syntax des alter exit state mapping Statements ist

```
alter [ existing ] exit state mapping mappingname
with map = ( statename { , signed_integer , statename } )
```

Beschreibung

Beschreibung

Das alter exit state mapping Statement definiert das Mapping der Exit Codes zu logischen Exit States. Die einfachste Form des Statements spezifiziert nur einen Exit State. Das bedeutet, dass der Job,ohne Rücksicht auf seinen Exit Code zu nehmen, diesen Exit State bei Beendigung bekommt. Komplexere Definitionen spezifizieren mehr als einen Exit State und mindestens eine Abgrenzung.

Ausgabe

Ausgabe

alter exit state profile

User Commands

alter exit state profile

Zweck

Das alter exit state profile Statement wird eingesetzt um Eigenschaften des spez- Zweck ifizierten Profiles zu ändern.

Syntax

Die Syntax des alter exit state profile Statements ist

Syntax

```
alter [ existing ] exit state profile profilename
with WITHITEM {, WITHITEM}

WITHITEM:
    default mapping = < none | mappingname >
    | force
    | state = ( ESP_STATE {, ESP_STATE} )

ESP_STATE:
    statename < final | restartable | pending > [ OPTION { OPTION} ]

OPTION:
    < unreachable | broken | batch default | dependency default >
```

Beschreibung

Das alter exit state profile Statement wird benutzt um Exit States an dem Profile zuzufügen oder zu löschen, sowie das default Exit State Mapping zu definieren. Für eine ausführliche Beschreibung der Optionen siehe das create exit state profile Statement auf Seite 90.

Beschreibung

force Die **force** option kennzeichnet die Exit State Profiles als invalid, das bedeutet nur, dass die Integrität noch geprüft werden muss. Nach einer erfolgreichen Überprüfung wird die Kennzeichnung gelöscht. Die Überprüfung wird beim Submitten einer Job Definition, welche die Exit State Profiles verwendet, durchgeführt. Das Ziel von dem **force** Flag ist, imstande zu sein mehrere Exit State Profiles, und vielleicht andere Objekte, zu ändern, ohne die Notwendigkeit eines konsistenten Zustands nach jeder Änderung.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

alter folder

alter folder

Zweck

Zweck

Das alter folder Statement wird eingesetzt um die Eigenschaften eines Folders zu ändern.

Syntax

Syntax

Die Syntax des alter folder Statements ist

```
alter [existing] folder folderpath
with WITHITEM {, WITHITEM}
WITHITEM:
     environment = < none | environmentname >
     group = groupname [ cascade ]
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     parameter = none
     parameter = ( parametername = string {, parametername = string} )
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
     use
     view
```

Beschreibung

Beschreibung

Das alter folder Statement ändert die Eigenschaften eines Folders. Für eine ausführliche Beschreibung der Optionen, siehe das create folder Statement auf Seite

Wenn das optionale Schlüsselwort **existing** spezifiziert ist, wird keine Fehlermeldung erzeugt wenn der spezifizierte Folder nicht existiert.

42 l alter folder

alter folder

User Commands

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

alter footprint

alter footprint

Zweck

Zweck

Das alter footprint Statement wird eingesetzt um die Eigenschaften des spezifizierten Footprints zu ändern.

Syntax

Syntax

Die Syntax des alter footprint Statements ist

```
alter [ existing ] footprint footprintname
with resource = ( REQUIREMENT {, REQUIREMENT} )

alter [ existing ] footprint footprintname
add resource = ( REQUIREMENT {, REQUIREMENT} )

alter [ existing ] footprint footprintname
delete resource = ( resourcepath {, resourcepath} )

REQUIREMENT:
ITEM { ITEM}

ITEM:
    amount = integer
    | < nokeep | keep | keep final >
    | resourcepath
```

Beschreibung

Beschreibung

Das alter footprint Kommando ändert die Liste der Resource-Anforderungen. Es gibt drei Formen des Statements.

- Die erste bestimmt alle Resource-Anforderungen.
- Die zweite fügt Resource-Anforderungen zu der Anforderungsliste zu.
- Die dritte Form entfernt Anforderungen aus der Liste.

Für eine ausführliche Beschreibung der Optionen, siehe das create footprint Statement auf Seite 95.

Ausgabe

Ausgabe

alter group

User Commands

alter group

Zweck

Das alter group Statement wird eingesetzt um die Zuordnung von Benutzer zu Zweck Gruppen zu ändern.

Syntax

Die Syntax des alter group Statements ist

Syntax

```
alter [ existing ] group groupname
with WITHITEM

alter [ existing ] group groupname
ADD_DELITEM {, ADD_DELITEM}

WITHITEM:
    user = none
    | user = ( username {, username} )

ADD_DELITEM:
    < add | delete > user = ( username {, username} )
```

Beschreibung

Das alter group Kommando wird verwendet um festzulegen welche Benutzer Beschreibung zu der Gruppe gehören. Es gibt zwei Formen des Statements:

- Die erste legt die Liste der Benutzer die zu der Gruppe gehören fest.
- Die zweite fügt Benutzer in die Gruppe zu oder löscht sie.

In allen Fällen wird es als Fehler betrachtet, Benutzer aus ihrer Default-Gruppe zu löschen.

Es ist nicht möglich Benutzer aus der Gruppe PUBLIC zu löschen.

Wenn ein Benutzer nicht zu einer Gruppe gehört, wird der Versuch den Benutzer aus dieser Gruppe zu löschen ignoriert.

Ist das Schlüsselwort **existing** spezifiziert, wird es nicht als Fehler betrachtet wenn die Gruppe nicht existiert.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

alter interval

alter interval

Zweck

Zweck

Das alter interval Statement wird eingesetzt um Eigenschaften des spezifizierten Intervalles zu ändern.

Syntax

Syntax

Die Syntax des alter interval Statements ist

```
alter [existing] interval intervalname
with WITHITEM {, WITHITEM}
WITHITEM:
     base = < none | period >
     duration = < none | period >
     embedded = < none | intervalname >
     endtime = < none | datetime >
     filter = none
     filter = ( intervalname {, intervalname} )
     < noinverse | inverse >
     selection = none
     selection = (IVAL_SELITEM {, IVAL_SELITEM})
     starttime = < none | datetime >
     synctime = datetime
     group = groupname
IVAL SELITEM:
< signed_integer | datetime | datetime - datetime >
```

Beschreibung

Beschreibung

Das alter interval Kommando wird benutzt um eine Intervalldefinition zu ändern. Für eine ausführliche Beschreibung der Optionen, siehe den create interval Statement auf Seite 98.

Ist das Schlüsselwort **existing** spezifiziert, wird es nicht als Fehler betrachtet, wenn der Intervall nicht existiert.

Ausgabe

Ausgabe

alter job

User Commands

alter job

Zweck

Das alter job Statement wird benutzt um Eigenschaften des spezifizierten Jobs zu Zweck ändern. Es wird von den Job Administratoren, Jobservern und von dem Job selbst benutzt.

Syntax

JOBSTATE:

Die Syntax des alter job Statements ist

Syntax

```
alter job jobid
with WITHITEM {, WITHITEM}
alter job
with WITHITEM {, WITHITEM}
WITHITEM:
     < suspend | suspend restrict | suspend local | suspend local restrict >
    cancel
     clear warning
     comment = string
     error text = string
     exec pid = pid
     exit code = signed_integer
     exit state = statename [ force ]
     ext pid = pid
     ignore resource = ( id {, id} )
     ignore dependency = ( jobid [ recursive ] {, jobid [ recursive ]} )
     kill
     nicevalue = signed_integer
     priority = integer
     renice = signed_integer
     rerun [recursive]
     resume
     < noresume | resume in period | resume at datetime >
     run = integer
     state = JOBSTATE
     timestamp = string
     warning = string
```

alter job

broken active
broken finished
dependency wait
error
finished
resource wait
running
started
starting
synchronize wait

Beschreibung

Beschreibung

Das alter job Kommando wird für mehrere Zwecke genutzt. Als erstes verwenden Jobserver dieses Kommando um den Ablauf eines Jobs zu dokumentieren. Alle Statuswechsel eines Jobs während der Zeit in der der Job innerhalb der Zuständigkeit eines Jobservers fällt, werden mittels des alter job Kommandos ausgeführt.

Zweitens werden einige Änderungen wie z. B. das Ignorieren von Abhängigkeiten oder Resourcen sowie das Ändern der Priorität eines Jobs manuell von einem Administrator ausgeführt.

Der Exit State eines Jobs in einem Pending State kann von dem Job selbst gesetzt werden, bzw. von einem Prozess welcher die Job ID und den Key des zu ändernden Jobs kennt.

cancel Die cancel Option wird benutzt um den adressierten Job und alle nicht Final Children zu canceln. Ein Job kann nur gecancelled werden wenn weder der Job selbst noch einer seiner Kinder aktiv ist.

Wenn ein Scheduling Entity von dem gecancelten Job abhängig ist, kann er unreachable werden. In diesem Fall erhält der abhängige Job nicht den im Exit State Profile definierten Unreachable Exit State, sondern wird in den Job Status "Unreachable" versetzt. Es ist Aufgabe des Operators diese Jobs nun mittels des Ignorierens von Abhängigkeiten wieder in den Job Status "Dependency Wait" zu versetzen, oder aber diese Jobs auch zu canceln.

Gecancelte Jobs werden wie Final Jobs ohne Exit State betrachtet. Das bedeutet, die Parents eines gecancelten Jobs werden final, ohne den Exit State des gecancelten Jobs zu berücksichtigen. Die abhängigen Jobs der Parents laufen in diesem Fall normal weiter.

Die cancel Option kann nur von Benutzern genutzt werden.

comment Die comment Option wird benutzt um eine Aktion zu dokumentieren oder um dem Job einen Comment zuzufügen. Comments können maximal 1024 Zeichen lang sein. Es kann eine beliebige Anzahl Comments für einen Job gespeichert werden.

Einige Comments werden automatisch gespeichert. Wenn z. B. ein Job einen Restartable State erreicht, wird ein Protokoll geschrieben, um diesen Fakt zu dokumentieren.

error text Die error text Option wird benutzt um Fehlerinformation zu einem Job zu schreiben. Dieses kann von dem verantwortlichen Jobserver oder einem Benutzer gemacht werden. Der Server kann diesen Text auch selbst schreiben. Diese Option wird normalerweise benutzt, wenn der Jobserver den entsprechenden Prozess nicht Starten kann. Mögliche Fälle sind die Unmöglichkeit zum definierten Working Directory zu wechseln, die Unauffindbarkeit des ausführbaren Programmes oder Fehler beim Öffnen des Error Logfiles.

exec pid Die exec pid Option wird ausschließlich von dem Jobserver benutzt um die Prozess ID des Kontroll Prozesses innerhalb des Servers zu setzen.

exit code Die exit code Option wird von dem Jobserver benutzt um dem Repository Server mitzuteilen mit welchem Exit Code sich ein Prozess beendet hat. Der Repository Server berechnet jetzt den zugehörigen Exit State aus dem verwendeten Exit State Mapping.

exit state Die Exit State Option wird von Jobs in einem Pending State benutzt, um ihren State auf einen anderen Wert zu setzen. Dies wird normalerweise ein Restartable oder Final State sein.

Alternativ dazu kann diese Option von Administratoren benutzt werden, um den State von einem nonfinal Job zu setzen.

Sofern das Force Flag nicht benutzt wird, sind die einzigen States die gesetzt werden können, die States welche, durch die Anwendung des Exit State Mappings auf irgendeinem Exit Code, theoretisch erreichbar sind. Der gesetzte State muss im Exit State Profile vorhanden sein.

ext pid Die ext pid Option wird ausschließlich von dem Jobserver genutzt, um die Prozess ID des gestarteten Benutzerprozesses zu setzen.

ignore resource Die ignore resource Option wird benutzt um einzelne Resource Requests aufzuheben. Die ignorierte Resource wird nicht mehr beantragt.

Wenn Parameter einer Resource referenziert werden, kann diese Resource nicht ignoriert werden.

Wenn ungültige ID's spezifiziert wurden, wird dies übergangen. Alle anderen spezifizierten Resources werden ignoriert. Ungültige ID's in diesem Kontext sind ID's von Resources die von dem Job nicht beantragt werden.

Das Ignorieren von Resources wird protokolliert.

ignore dependency Die ignore dependency Option wird benutzt um definierte Dependencies zu ignorieren. Wenn das **recursive** flag benutzt wird, ignorieren nicht nur der Job oder Batch selbst, sondern auch seine Kinder die Dependencies.

kill Die kill Option wird benutzt um den definierten Kill Job zu submitten. Wenn kein Kill Job definiert ist, ist es nicht möglich den Job vom BICsuite aus erzwungenermaßen zu terminieren. Natürlich muss der Job aktiv sein, das bedeutet, der Jobstate muss **running**, **killed** oder **broken_active** sein. Die letzten beiden States sind keine regulären Fälle.

Wenn ein Kill Job submitted wurde, ist der Jobstate to_kill. Nachdem der Kill Job beendet wurde, wird der Jobstate des killed Jobs in den State killed gesetzt, es sei denn er ist beendet, dann wird der Jobstate finished oder final sein. Das bedeutet, der Job mit dem Jobstate killed bedeutet, dass er immer noch running ist und dass mindestens ein Versuch gemacht wurde, den Job zu terminieren.

nicevalue Die nicevalue Option wird benutzt um die Priorität oder den Nicevalue eines Jobs oder Batches und allen ihren Children zu ändern. Hat ein Child mehrere Parents, kann eine Änderung, muss aber nicht, in dem Nicevalue von einem der Parents Auswirkungen auf die Priorität des Childs haben. In dem Fall das es mehrere Parents gibt wird das maximale Nicevalue gesucht.

Also, wenn Job C drei Parents P1, P2 und P3 hat und P1 setzt einen Nicevalue von 0, P2 einen von 10 und P3 einen von -10, ist der effektive Nicevalue -10. (Umso niederiger der Nicevalue, umso besser). Wenn der Nicevalue von P2 auf -5 geändert wird, passiert nichts, weil die -10 von P3 besser als -5 ist. Wenn jetzt der Nicevalue von P3 auf 0 sinkt, wird die neue effektive Nicevalue für Job C -5.

Die Nicevalues können Werte zwischen -100 und 100 haben. Werte die diese Spanne übersteigen, werden stillschweigend angepasst.

priority Die priority Option wird benutzt, um die (statische) Priorität eines Jobs zu ändern. Weil Batches und Milestones nicht ausgeführt werden, haben Prioritäten keine Bedeutung für sie.

Ein Wechsel der Priorität betrifft nur den geänderten Job. Gültige Werte liegen zwischen 0 und 100. Dabei korrespondiert 100 mit der niedrigsten Priorität und 0 mit der höchsten Priorität.

Bei der Berechnung der dynamischen Priorität eines Jobs, startet der Scheduler mit der statischen Priorität und passt dies, entsprechend der Zeit in der der Job schon wartet, an. Wenn mehr als ein Job die gleiche dynamische Priorität hat, wird der Job mit der niedrigsten Job ID als erster gescheduled.

renice Die renice Option gleicht der nicevalue Option mit dem Unterschied, das die renice Option relativ arbeitet, während die nicevalue Option absolut arbeitet. Wenn einige Batches einen Nicevalue von 10 haben bewirkt eine renice von -5,

dass die Nicevalue auf 5 zunimmt. (Zunahme, weil je niedriger die Nummer, desto höher die Piorität)

rerun Die rerun Option wird benutzt um einen Job in einem Restartable State neu zu starten. Der Versuch einen Job, der nicht restartable ist, neu zu starten, führt zu einer Fehlermeldung. Ein Job ist restartable, wenn er in einem Restartable State oder in einem **error** oder **broken_finished** Jobstate ist.

Wenn das **recursive** flag spezifiziert ist, wird der Job selbst und alle direkten und indirekten Children, die in einem Restartable State sind, neu gestartet. Wenn der Job selbst Final ist, wird das in dem Fall nicht als Fehler betrachtet. Es ist also möglich Batches rekursiv neu zu starten.

resume Die resume Option wird benutzt um einen Suspended Job oder Batch zu reaktivieren. Es gibt dabei zwei Möglichkeiten. Erstens kann der suspended Job oder Batch sofort reaktiviert werden, und zweitens kann eine Verzögerung eingestellt werden.

Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll.

Für die Spezifikation einer Zeit siehe auch die Übersicht auf Seite 6.

Die resume Option kann zusammen mit der suspend Option verwendet werden. Dabei wird der Job suspended und nach der (bzw. zur) spezifizierten Zeit wieder resumed.

run Die run Option wird von dem Jobserver benutzt zwecks der Sicherstellung, dass der geänderte Job mit der aktuellen Version übereinstimmt.

Theoretisch ist es möglich, dass nachdem ein Job von einem Jobserver gestartet wurde, der Computer abstürzt. Um die Arbeit zu erledigen wird der Job mittels eines manuellen Eingriffs, von einem anderen Jobserver, neu gestartet. Nach dem hochfahren des ersten Systems, kann der Jobserver versuchen den Jobstate nach broken_finished zu ändern, ohne über das Geschehen nach dem Absturz bescheid zu wissen. Das Benutzen der run Option, verhindert nun das fälschlich Setzen des Status.

state Die state Option wird hauptsächlich von Jobservern benutzt, sie kann aber auch von Administratoren benutzt werden. Es wird nicht empfohlen das so zu machen, es sei denn Sie wissen genau was Sie tun.

Die übliche Prozedur ist, das der Jobserver den State eines Jobs von **starting** nach **started**, von **started** nach **running** und von **running** nach **finished** setzt. Im Falle eines Absturzes oder anderen Problemen ist es möglich dass der Jobserver einen Job in einen **broken_active** oder **broken_finished** State setzt. Das bedeutet, der Exit Code von dem Prozess steht nicht zur Verfügung und der Exit State muss manuell gesetzt werden.

alter job

suspend Die suspend Option wird benutzt um einen Batch oder Job zu suspendieren. Sie arbeitet immer rekursiv. Wenn ein Parent Suspended ist, sind auch alle Children suspended. Die resume Option wird benutzt um die Situation zu umzukehren.

Die **restrict** Option bewirkt, dass nur Benutzer der ADMIN Gruppe die Suspendierung wieder aufheben können.

timestamp Die timestamp Option wird von dem Jobserver benutzt um die Timestamps der Statuswechsel zu setzen, gemäß der lokalen Zeit aus Sicht des Jobservers.

Ausgabe

Ausgabe

alter job definition

Zweck

Das alter job definition Statement wird eingesetzt um die Eigenschaften der spe- Zweck zifizierten Job Definition zu ändern.

Syntax

Die Syntax des alter job definition Statements ist

Syntax

```
alter [ existing ] job definition folderpath . jobname
with withitem {, withitem}
alter [ existing ] job definition folderpath . jobname
AJD_ADD_DEL_ITEM {, AJD_ADD_DEL_ITEM}
WITHITEM:
     children = none
    children = ( JOB_CHILDDEF {, JOB_CHILDDEF} )
     dependency mode = < all | any >
     environment = environmentname
     errlog = < none | filespec [ < notrunc | trunc > ] >
     footprint = < none | footprintname >
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     kill program = < none | string >
     logfile = < none | filespec [ < notrunc | trunc > ] >
     mapping = < none | mappingname >
     < nomaster | master >
     nicevalue = < none | signed_integer >
     parameter = none
     parameter = ( JOB_PARAMETER {, JOB_PARAMETER} )
     priority = < none | signed_integer >
     profile = profilename
     required = none
     required = ( JOB_REQUIRED {, JOB_REQUIRED} )
     rerun program = < none | string >
     resource = none
     resource = ( REQUIREMENT {, REQUIREMENT} )
     < noresume | resume in period | resume at datetime >
     runtime = integer
     runtime final = integer
```

```
run program = < none | string >
     < nosuspend | suspend >
     timeout = none
     timeout = period state statename
     type = < job | milestone | batch >
     group = groupname
     workdir = < none | string >
AJD_ADD_DEL_ITEM:
     add [ or alter ] children = ( JOB_CHILDDEF {, JOB_CHILDDEF} )
     add | or alter | parameter = ( JOB_PARAMETER {, JOB_PARAMETER} )
     add [ or alter ] required = ( JOB_REQUIRED {, JOB_REQUIRED} )
     add [ or alter ] resource = ( REQUIREMENT {, REQUIREMENT} )
     alter [ existing ] children = ( JOB_CHILDDEF {, JOB_CHILDDEF} )
     alter [ existing ] parameter = ( JOB_PARAMETER {, JOB_PARAMETER} )
     alter [ existing ] required = ( JOB_REQUIRED {, JOB_REQUIRED} )
     alter [ existing ] resource = ( REQUIREMENT {, REQUIREMENT} )
     delete [ existing ] children = ( folderpath {, folderpath} )
     delete [ existing ] parameter = ( parmlist )
     delete [ existing ] required = ( folderpath {, folderpath} )
     delete [ existing ] resource = ( resourcepath {, resourcepath} )
JOB CHILDDEF:
JCD_ITEM { JCD_ITEM}
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
     use
     view
JOB_PARAMETER:
parametername < [ JP_WITHITEM ] [ default = string ] | JP_NONDEFWITH >
[local] [ < export = parametername | export = none > ]
```

```
JOB_REQUIRED:
JRQ_ITEM { JRQ_ITEM}
REOUIREMENT:
JRD_ITEM { JRD_ITEM}
JCD_ITEM:
     alias = < none | aliasname >
    < enable | disable >
    folderpath.jobname
    ignore dependency = none
     ignore dependency = ( dependencyname {, dependencyname} )
     < childsuspend | suspend | nosuspend >
     merge mode = < nomerge | merge local | merge global | failure >
     nicevalue = < none | signed_integer >
     priority = < none | signed_integer >
     < noresume | resume in period | resume at datetime >
     < static | dynamic >
JP_WITHITEM:
     import
    parameter
    reference child folderpath ( parametername )
    reference folderpath ( parametername )
     reference resource resourcepath (parametername)
    result
JP_NONDEFWITH:
     constant = string
  JP_AGGFUNCTION ( parametername )
JRQ_ITEM:
     dependency dependencyname
    folderpath.jobname
    mode = < all final | job final >
    state = none
    state = ( JRQ_REQ_STATE {, JRQ_REQ_STATE} )
     state = all reachable
     state = default
     state = unreachable
```

alter job definition

```
unresolved = < error | ignore | suspend | defer >
JRD_ITEM:
     amount = integer
     expired = < none | signed_period >
     < nokeep | keep | keep final >
     lockmode = LOCKMODE
     nosticky
     resourcepath
     state = none
     state = ( statename {, statename} )
     state mapping = < none | rsmname >
     sticky
   [ ( < identifier | folderpath | identifier , folderpath | folderpath , identifier > ) ]
JP_AGGFUNCTION:
     avg
     count
     max
     min
     sum
JRQ_REQ_STATE:
statename [ < condition = string | condition = none > ]
LOCKMODE:
     n
     \mathbf{s}
     sc
     \mathbf{s}\mathbf{x}
     X
```

Beschreibung

Beschreibung

Das alter job definition Kommando kennt zwei verschiedene Varianten.

- Die erste ähnelt dem create job definition Statement und wird benutzt um die Job Definition erneut zu definieren. Alle betroffenen Optionen werden überschrieben. Alle nichtadressierten Optionen verbleiben wo sie sind.
- Die zweite Variante wird benutzt um Einträge aus den Listen der Children, Resource-Anforderungen, Abhängigkeiten oder Parameter hinzuzufügen, zu ändern oder zu löschen.

Die Optionen werden ausführlich in dem create job definition Kommando auf Seite 102 beschrieben. Dieses gilt auch für die Optionen in Child-, Resourceanforderungs-, Dependency- und Parameter-Definitionen.

Wird das **existing** Schlüsselwort verwendet, wird kein Fehler erzeugt wenn die adressierte Job Definition nicht existiert. Ist das **existing** Schlüsselwort in Benutzung während der Löschung oder Änderung der Listeneinträge gilt auch für sie dasselbe.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

alter named resource

alter named resource

Zweck

Zweck

Das alter named resource Statement wird eingesetzt um die Eigenschaften einer Named Resource zu ändern.

Syntax

Syntax

Die Syntax des alter named resource Statements ist

```
alter [ existing ] named resource resourcepath
with WITHITEM {, WITHITEM}
WITHITEM:
     group = groupname [ cascade ]
    inherit grant = none
    inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     parameter = none
     parameter = ( PARAMETER {, PARAMETER} )
     state profile = < none | rspname >
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
     use
     view
PARAMETER:
     parametername constant = string
   | parametername local constant [ = string ]
  | parametername parameter [ = string ]
```

Ausgabe

alter named resource

User Commands

Beschreibung

Das alter named resource Statement wird verwendet um Eigenschaften der Named *Beschreibung* Resources zu ändern. Für eine ausführliche Beschreibung der Optionen siehe die Beschreibung des create named resource Statements auf der Seite 121. Wenn das Schlüsselwort **existing** spezifiziert wird, führt der Versuch eine nicht existierende Named Resource zu ändern nicht zu einem Fehler.

Ausgabe

alter resource

alter resource

Zweck

Zweck

Das alter resource Statement wird eingesetzt um die Eigenschaften von Resourcen zu ändern.

Syntax

Syntax

Die Syntax des alter resource Statements ist

```
alter [ existing ] RESOURCE_URL
with WITHITEM {, WITHITEM}
RESOURCE_URL:
     resource resourcepath in folderpath
    resource resourcepath in serverpath
WITHITEM:
     amount = < infinite | integer >
     < online | offline >
     parameter = none
     parameter = ( PARAMETER {, PARAMETER} )
     requestable amount = < infinite | integer >
     state = statename
     touch [ = datetime ]
     group = groupname
PARAMETER:
parametername = < string | default >
```

Beschreibung

Beschreibung

Das alter resource Statement wird verwendet um die Eigenschaften von Resourcen zu ändern. Für eine ausführliche Beschreibung der Optionen siehe die Beschreibung des Create Resource Statements auf Seite 124.

Wenn das **existing** Schlüsselwort spezifiziert ist, wird der Versuch eine nicht existierende Resource zu ändern nicht zu einem Fehler führen.

Ausgabe

Ausgabe

alter resource state mapping

Zweck

Das alter resource state mapping Statement wird eingesetzt um die Eigenschaften Zweck eines Mappings zu ändern.

Syntax

Die Syntax des alter resource state mapping Statements ist

Syntax

```
alter [existing] resource state mapping mappingname
with map = ( WITHITEM {, WITHITEM} )
```

WITHITEM:

statename maps < statename | any > to statename

Beschreibung

Das alter resource state mapping Statement wird verwendet um die Eigenschaften Beschreibung des Resource State Mappings zu ändern. Für eine ausführliche Beschreibung der Optionen siehe die Beschreibung des create resource state mapping Statements auf Seite 129.

Wenn das existing Schlüsselwort spezifiziert ist, wird der Versuch ein nicht existierendes Resource State Mapping zu ändern nicht zu einem Fehler führen.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter resource state profile

alter resource state profile

Zweck

Zweck

Das alter resource state profile Statement wird eingesetzt die Eigenschaften des spezifizierten Resource State Profiles zu ändern.

Syntax

Syntax

Die Syntax des alter resource state profile Statements ist

```
alter [ existing ] resource state profile profilename
with WITHITEM {, WITHITEM}

WITHITEM:
    initial state = statename
    | state = ( statename {, statename} )
```

Beschreibung

Beschreibung

Das alter resource state profile Statement wird werwendet um Eigenschaften der Resource State Profiles zu ändern. Für eine ausführliche Beschreibung der Optionen siehe die Beschreibung der Resource State Profile Statements auf Seite 130. Wenn das **existing** Schlüsselwort spezifiziert ist, wird der Versuch ein nicht existierendes Resource State Profile zu ändern, nicht zu einem Fehler führen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter schedule

User Commands

alter schedule

Zweck

Das alter schedule Statement wird eingesetzt um die Eigenschaften des spezi- Zweck fizierten Zeitplans zu ändern.

Syntax

Die Syntax des alter schedule Statements ist

Syntax

```
alter [existing] schedule schedulepath
with WITHITEM {, WITHITEM}
WITHITEM:
     < active | inactive >
    inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     interval = < none | intervalname >
     time zone = string
     group = groupname
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
     use
```

Beschreibung

view

Das alter schedule Statement wird verwendet um die Eigenschaften eines Schedules zu ändern. Für eine ausführliche Beschreibung der Optionen des create schedule Statements siehe Seite 131.

Beschreibung

Wenn das **existing** Schlüsselwort spezifiziert ist, wird der Versuch ein nicht existierendes schedule zu ändern, nicht zu einem Fehler führen.

alter schedule

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter scheduled event

Zweck

Das alter scheduled event Statement wird eingesetzt um Eigenschaften des spez- *Zweck* ifizierten Scheduled Events zu ändern.

Syntax

Die Syntax des alter scheduled event Statements ist

Syntax

Beschreibung

Das alter scheduled event Statement wird verwendet um die Eigenschaften eines Scheduled Events zu ändern. Für eine ausführliche Beschreibung der Optionen des create scheduled event Statements siehe Seite 133.

Beschreibung

Wenn das **existing** Schlüsselwort spezifiziert ist, wird der Versuch ein nicht existierendes scheduled Event zu ändern, nicht zu einem Fehler führen.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter scope

alter scope

Zweck

Zweck

Das alter scope Statement wird eingesetzt um die Eigenschaften eines spezifizierten Scopes zu ändern.

Syntax

Syntax

Die Syntax des alter scope Statements ist

```
alter [existing] < scope serverpath | job server serverpath >
with is withitem {, is withitem}
alter [ existing ] job server
with < fatal | nonfatal > error text = string
alter [existing] job server
with dynamic PARAMETERS
JS_WITHITEM:
     config = none
     config = ( CONFIGITEM {, CONFIGITEM} )
     < enable | disable >
     error text = < none | string >
     group = groupname [ cascade ]
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     node = nodename
     parameter = none
     parameter = ( PARAMETERITEM {, PARAMETERITEM} )
     password = string
     rawpassword = string [ salt = string ]
PARAMETERS:
     parameter = none
     parameter = ( PARAMETERSPEC {, PARAMETERSPEC} )
CONFIGITEM:
     parametername = none
  parametername = ( PARAMETERSPEC {, PARAMETERSPEC} )
```

alter scope

User Commands

```
| parametername = < string | number >
PRIVILEGE:
    create content
    drop
    edit
    execute
    monitor
    operate
    resource
    submit
    use
    view
PARAMETERITEM:
    parametername = dynamic
  | parametername = < string | number >
PARAMETERSPEC:
parametername = < string | number >
```

Beschreibung

Das alter scope Kommando ist ein Benutzerkommando. Dieses Kommando wird Beschreibung benutzt um die Konfiguration oder andere Eigenschaften eines Scopes zu ändern.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter server

alter server

Zweck

Zweck

Das alter server Statement wird eingesetzt um die Benutzerverbindungen zu aktivieren oder deaktivieren oder um den trace level fest zu legen.

Syntax

Syntax

Die Syntax des alter server Statements ist

alter server with < enable | disable > connect

alter server with schedule

alter server with trace level = integer

Beschreibung

Beschreibung

Das alter server Kommando kann verwendet werden um die Möglichkeit sich mit dem Server zu verbinden ein- und auszuschalten. Wurde die Möglichkeit sich mit dem Server zu verbinden ausgeschaltet, kann sich nur der Benutzer "System" verbinden.

Das alter server Kommando wird ebenso verwendet um die prokotollierten Typen von Servernachrichten zu definieren. Die folgenden Informationstypen werden definiert:

Type	Bedeutung		
Fatal	Ein fataler Fehler ist aufgetreten. Der Server		
	wird runtergefahren.		
Error	Ein Fehler ist aufgetreten.		
Info	Eine wichtige informative Nachricht, die nicht		
	Aufgrund eines Fehlers geschrieben wird.		
Warning	Eine Warnung.		
Message	Eine informative Nachricht.		
Debug	Meldungen die für die Fehlersuche benutzt		
<u> </u>	werden können.		

Fatal-Nachrichten, Error-Nachrichten und Info-Nachrichten werden immer ins Server Logfile geschrieben. Warnings werden geschrieben, wenn der trace level 1 oder höher ist, normale Messages werden mit einem trace level von 2 oder höher geschrieben. Debug-Nachrichten liefern sehr viel Output und werden ausgegeben wenn der trace level 3 ist.

alter server

User Commands

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter session

alter session

Zweck

Zweck

Das alter session Statement wird eingesetzt um den genutzten Protokoll, den session timeout Wert oder den trace level für die spezifizierte Session zu setzen.

Syntax

Syntax

Die Syntax des alter session Statements ist

```
alter session [ sid ]
with WITHITEM {, WITHITEM}

WITHITEM:
    command = ( sdms-command )
    | protocol = PROTOCOL
    | session = string
    | timeout = integer
    | < trace | notrace >
    | trace level = integer

PROTOCOL:
    json
    | line
    | perl
    | python
    | serial
```

Beschreibung

xml

Beschreibung

Das alter session Kommando kann verwendet werden um die trace ein- und auszuschalten. Ist die trace eingeschaltet, werden alle abgesetzten Befehle ins Logfile protokolliert. Desweiteren kann ein Kommunikationsprotokoll gewählt werden. Eine Übersicht der derzeitig definierten Protokolle wird in der untenstehenden Tabelle angezeigt.

70 I

alter session User Commands

Protokoll	Bedeutung
Line	Reines ASCII Output
Perl	Die Ausgabe wird als peil Struktur angeboten,
	die mittels eval auf einfacher Weise dem perl-
	script bekannt gemacht werden kann.
Python	Wie perl, nur handelt es sich hier um eine
	python Struktur.
Serial	Serialisierte Java Objekte.
Xml	Eine xml Strucktur wird ausgegeben.

Als letzte Möglichkeit kann der timeout Parameter für die Session festgelegt werden. Eine timeout von 0 bedeutet, dass kein timeout aktiv ist. Jede Zahl größer als 0 gibt die Anzahl Sekunden an, nach der eine Session automatisch disconnected wird.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter trigger

alter trigger

Zweck

Zweck

Das alter trigger Statement wird eingesetzt um Eigenschaften des spezifizierten Triggers zu ändern.

Syntax

Syntax

Die Syntax des alter trigger Statements ist

```
alter [ existing ] trigger triggername on TRIGGEROBJECT [ < noinverse |
inverse > ]
with WITHITEM {, WITHITEM}
TRIGGEROBJECT:
     resource resourcepath in folderpath
    job definition folderpath
     object monitor objecttypename
    resource resourcepath in serverpath
WITHITEM:
     < active | inactive >
    < nowarn | warn >
    limit state = < none | statename >
     < nomaster | master >
     rerun
     < noresume | resume in period | resume at datetime >
     state = none
     state = ( < statename {, statename} |</pre>
     CT_RSCSTATUSITEM {, CT_RSCSTATUSITEM} > )
     submit after folderpath
     submit folderpath
     submitcount = integer
     < nosuspend | suspend >
     [type = ] CT_TRIGGERTYPE
     group = groupname
CT_EVENT:
< create | change | delete >
CT_RSCSTATUSITEM:
```

alter trigger

User Commands

< statename any | statename statename | any statename >

CT_TRIGGERTYPE:

before final immediate local immediate merge warning

Beschreibung

Das alter trigger Kommando wird benutzt um die Eigenschaften eines definierten Beschreibung Triggers zu ändern. Wenn das **existing** Schlüsselwort spezifiziert ist, führt das Ändern eines nicht existierenden Triggers nicht zu einem Fehler. Für eine ausführliche Beschreibung der Optionen siehe das create trigger Statement auf Seite 138.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

alter user

alter user

Zweck

Zweck

Das alter user Statement wird eingesetzt um Eigenschaften des spezifizierten Benutzers zu ändern.

Syntax

Syntax

Die Syntax des alter user Statements ist

```
alter [ existing ] user username
with WITHITEM {, WITHITEM}

alter [ existing ] user username
ADD_DELITEM {, ADD_DELITEM}

WITHITEM:
    default group = groupname
    | < enable | disable >
    | group = (groupname {, groupname})
    | password = string
    | rawpassword = string [ salt = string ]

ADD_DELITEM:
    < add | delete > group = (groupname {, groupname})

ADD_DELITEM:
    < add | delete > group = (groupname {, groupname})
```

Beschreibung

Beschreibung

Das alter user Kommando wird verwendet um die Eigenschaften eines definierten Users zu ändern. Wenn das **existing** Schlüsselwort spezifiziert ist, führt der Versuch einen nicht existierenden User zu ändern nicht zu einem Fehler.

Für eine ausführliche Beschreibung der Optionen siehe das create user Statement auf Seite 147.

Die zweite Form des Statements wird benutzt um den User von den spezifizierten Gruppen zu löschen oder zuzufügen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

74 l alter user

4. connect commands

connect

connect

Zweck

Zweck

Das connect Statement wird eingesetzt um Benutzer von dem Server authentifizieren zu lassen.

Syntax

Syntax

Die Syntax des connect Statements ist

connect *username* **identified** by *string* [**with** WITHITEM {, WITHITEM}]

```
WITHITEM:
```

```
command = ( sdms-command )
| protocol = PROTOCOL
| session = string
| timeout = integer
| < trace | notrace >
| trace level = integer
```

PROTOCOL:

json | line | perl

python serial

xml

Beschreibung

Beschreibung

Das connect Kommando wird benutzt um den verbundenen Prozess beim Server zu authentifizieren. Es kann wahlweise ein Kommunikationsprotokoll spezifiziert werden. Das default protocol ist **line**.

Das ausgewählte Protokoll definiert die Form des Outputs. Alle Protokolle, außer serial, liefern ASCII Output. Das Protokoll serial liefert einen serialized Java Objekt zurück.

Beim Connect kann gleich ein auszuführendes Kommando mitgegeben werden. Als Output des Connect Kommandos wird in dem Fall der Output des mitgegebenen Kommandos genutzt. Falls das Kommando fehl schlägt, der Connect aber gültig war, bleibt die Connection bestehen.

Im folgenden ist für alle Protokolle, außer für das **serial** Protokoll, ein Beispiel gegeben.

76 I connect

Das line protocol Das line protocol liefert nur einen ASCII Text als Ergebnis von einem Kommando zurück.

```
connect donald identified by 'duck' with protocol = line;
Connect
CONNECT_TIME : 19 Jan 2005 11:12:43 GMT
Connected
SDMS>
```

Das XML protocol Das XML protocol liefert eine XML-Struktur als Ergebnis eines Kommandos zurück.

```
connect donald identified by 'duck' with protocol = xml;
<OUTPUT>
<DATA>
<TITLE>Connect</TITLE>
<RECORD>
<CONNECT_TIME>19 Jan 2005 11:15:16 GMT</CONNECT_TIME></RECORD>
</DATA>
<FEEDBACK>Connected</FEEDBACK>
</OUTPUT>
```

Das python protocol Das python protocol liefert eine python-Struktur, welche durch die python eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = python;
{
'DATA' :
{
'TITLE' : 'Connect',
'DESC' : [
'CONNECT_TIME'
],
'RECORD' : {
'CONNECT_TIME' : '19 Jan 2005 11:16:08 GMT'}
}
,'FEEDBACK' : 'Connected'
}
```

connect I 77

connect

Das perl protocol Das perl protocol liefert eine perl Struktur, welche durch die perl eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = perl;
{
'DATA' => {
   'TITLE' => 'Connect',
   'DESC' => [
   'CONNECT_TIME' |
],
   'RECORD' => {
   'CONNECT_TIME' => '19 Jan 2005 11:19:19 GMT'}
}, 'FEEDBACK' => 'Connected'
}
```

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

78 I connect

5. copy commands

copy folder

copy folder

Zweck

Zweck

Das copy folder Statement wird eingesetzt um einen oder mehreren Folder und/oder Job Definitionen mitsamt seines Inhalts an eine anderen Stelle in der Ordnerhierarchie zu kopieren.

Syntax

Syntax

Die Syntax des copy folder Statements ist

```
copy FOLDER_OR_JOB {, FOLDER_OR_JOB} to folderpath
copy FOLDER_OR_JOB {, FOLDER_OR_JOB} to foldername
FOLDER_OR_JOB:
[ < folder folderpath | job definition folderpath > ]
```

Beschreibung

Beschreibung

Wenn ein Folder kopiert wurde, wird jedes Objekt das der Folder enthält mitkopiert. Wenn Beziehungen zwischen Objekte bestehen, welche durch eine copy folder Operation kopiert wurden, wie z. B. Abhängigkeiten, Kinder, Trigger etc., werden diese entsprechend geändert und den resultierenden Objekten von der Kopie zugeordnet.

Wenn z. B. ein Ordner SYSTEM.X.F. zwei Jobs A und B enthält mit SYSTEM.X.F.B abhängig von SYSTEM.X.F.A, in den Ordner SYSTEM.Y kopiert wird, wird der neu angelegte Job SYSTEM.Y.F.B von dem neu angelegten Job SYSTEM.Y.F.A abhängig sein.

Beachten Sie das wenn die Jobs mit einem copy job definition Kommando kopiert wurden, der neue Job SYSTEM.Y.F.B immer noch vom SYSTEM.X.F.A abhängig wäre. Es kann sein, das es nicht der Ansicht des Users entspricht.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

copy named resource

Zweck

Das copy named resource Statement wird eingesetzt um eine Named Resource Zweck in eine andere Kategorie zu kopieren.

Syntax

Die Syntax des copy named resource Statements ist

Syntax

Beschreibung

copy named resource resourcepath to resourcepath

copy named resource resourcepath to resourcename

Beschreibung

Das copy named resource Kommando wird benutzt um eine Kopie von einer Named Resource oder einer ganzen Kategorie zu machen.

Wenn der spezifizierte target resourcepath bereits als Kategorie existiert, wird eine Named Resource oder Kategorie mit dem selben Namen wie der source Objekt, innerhalb dieser Kategorie angelegt.

Wenn das spezifizierte target resourcepath bereits als Named Resource existiert, wird dies als Fehler betrachtet.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

copy scope

copy scope

Zweck

Zweck

Das copy scope Statement wird eingesetzt um um einen Scope mitsamt seines Inhalts an einer anderen Stelle in der Scope-Hierarchie zu kopieren.

Syntax

Syntax

Die Syntax des copy scope Statements ist

copy < **scope** *serverpath* | **job server** *serverpath* > **to** *serverpath*

copy < **scope** *serverpath* | **job server** *serverpath* > **to** *scopename*

Beschreibung

Beschreibung

Das copy named resource Kommando wird benutzt um eine Kopie von ganzen Scopes zu machen. Diese Kopie umfasst auch Resource- und Parameter-Definitionen. Wenn der spezifizierte target serverpath bereits als Scope existiert, wird ein Scope mit dem selben Namen wie das source Objekt, innerhalb dieses Scopes angelegt. Wenn das spezifizierte target Serverpath bereits als Jobserver existiert, wird dies als Fehler betrachtet.

Da ein Jobserver nur als eine spezielle Art von Scope angesehen wird, ist es möglich Jobserver mit diesem Kommando zu Kopieren. In diesem Fall ist dieses Kommando identisch mit dem Copy Jobserver Kommando.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

6. create commands

create comment

create comment

Zweck

Zweck

Das create comment Statement wird eingesetzt um einen Kommentar zu dem spezifizierten Objekt anzulegen.

Syntax

Syntax

Die Syntax des create comment Statements ist

```
create [ or alter ] comment on OBJECTURL
with CC_WITHITEM
```

OBJECTURL:

```
distribution distributionname for pool resourcepath in serverpath
environment environmentname
exit state definition statename
exit state mapping mappingname
exit state profile profilename
event eventname
resource resourcepath in folderpath
folder folderpath
footprint footprintname
group groupname
interval intervalname
job definition folderpath
job jobid
named resource resourcepath
parameter parametername of PARAM_LOC
resource state definition statename
resource state mapping mappingname
resource state profile profilename
scheduled event schedulepath . eventname
schedule schedulepath
resource resourcepath in serverpath
< scope serverpath | job server serverpath >
```

```
CC_WITHITEM:
```

user username

CC_TEXTITEM {, CC_TEXTITEM}

trigger triggername on TRIGGEROBJECT [< noinverse | inverse >]

create comment

User Commands

```
PARAM_LOC:
    folder folderpath
    | job definition folderpath
    | < scope serverpath | job server serverpath >

TRIGGEROBJECT:
    resource resourcepath in folderpath
    | job definition folderpath
    | object monitor objecttypename
    | resource resourcepath in serverpath

CC_TEXTITEM:
    tag = < none | string > , text = string
    | text = string
```

Beschreibung

Das create comment Statement wird benutzt um die Kurzbeschreibung bzw. die URL der Beschreibung, des zu kommentierenden Objektes zu erstellen. Das optionale Schlüsselwort **or alter** wird benutzt um den Kommentar, wenn einer existiert, zu aktualisieren. Wenn es nicht spezifiziert ist, führt das Vorhandensein eines Kommentars zu einem Fehler.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create environment

create environment

Zweck

Zweck

Das create environment Statement wird eingesetzt um eine Anzahl von Static Named Resources, welche in den Scopes, in dem ein Job laufen will, gebraucht werden, zu definieren.

Syntax

Syntax

Die Syntax des create environment Statements ist

```
create [ or alter ] environment environmentname [ with
ENV_WITH_ITEM ]

ENV_WITH_ITEM:
    resource = none
    | resource = ( ENV_RESOURCE {, ENV_RESOURCE} )

ENV_RESOURCE:
    resourcepath [ < condition = string | condition = none > ]
```

Beschreibung

Beschreibung

Das create environment Statement wird benutzt um eine Reihe von Static Resource Requests festzulegen, welche die notwendige Umgebung, die ein Job braucht, beschreiben. Da die Environments nicht von normalen Benutzern angelegt werden können und Jobs den Environment, den sie zum Ablauf benötigen, beschreiben müssen, können Environments benutzt werden um Jobs zu zwingen einen bestimmten Jobserver zu benutzen.

Resources Die Resources Klausel wird benutzt um die Required (Static) Resources zu spezifizieren. Spezifizierte Resourcen, welche nicht static sind, führen zu einem Fehler. Da nur statische Resourcen spezifiziert werden, werden keine weiteren Informationen benötigt. Es ist zulässig einen leeren Environment (ein Environment ohne Resource-Anforderungen) zu spezifizieren. Dies wird nicht empfohlen, da es ein Verlust an Kontrolle bedeutet.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create event

User Commands

create event

Zweck

Das create event Statement wird eingesetzt um eine Aktion, die von dem Time Zweck Scheduling Modul ausgeführt wird, zu definieren.

Syntax

Die Syntax des create event Statements ist

Syntax

```
create [ or alter ] event eventname
with EVENT_WITHITEM {, EVENT_WITHITEM}

EVENT_WITHITEM:
    action =
    submit folderpath [ with parameter = ( PARAM {, PARAM} ) ]
    | group = groupname

PARAM:
parametername = < string | number >
```

Beschreibung

Das create event Statement wird benutzt um eine Aktion, die vom Time Scheduling System scheduled werden kann, zu definieren. Die definierte Aktion ist die Submission eines Master submittable Jobs oder Batches.

Beschreibung

action Der submit Teil von dem Statement ist eine eingeschränkte Form des Submit-Kommandos (Siehe Seite 368).

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create exit state definition

create exit state definition

Zweck

Zweck

Das create exit state definition Statement wird eingesetzt um einen symbolischen Namen für den State eines Jobs zu erstellen.

Syntax

Syntax

Die Syntax des create exit state definition Statements ist

create [**or alter**] **exit state definition** *statename*

Beschreibung

Beschreibung

Das create exit state definition Statement wird benutzt um einen symbolischen Namen für den Exit State eines Jobs, Milestones oder Batches zu definieren. Das optionale Schlüsselwort or alter wird benutzt um das Auftreten von Fehlermeldungen und das Abbrechen der laufenden Transaktion, wenn eine Exit State Definition bereits existiert, zu verhindern. Das ist in Kombination mit Multicommands besonders nützlich. Wenn es nicht spezifiziert ist, führt das existieren einer Exit State Definition mit dem spezifizierten Namen zu einem Fehler.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Beispiel

Beispiel

In den folgenden Beispielen wurden symbolische Namen für Job States erstellt.

```
create exit state definition erfolg;
create exit state definition fehler;
create exit state definition erreicht;
create exit state definition warnung;
create exit state definition warten;
create exit state definition ueberspringen;
create exit state definition unerreichbar;
```

create exit state mapping

Zweck

Das create exit state mapping Statement wird eingesetzt um ein Mapping zwis-Zweck chen dem numerischen Exit Code eines Prozesses und einem symbolischen Exit State zu erstellen.

Syntax

Die Syntax des create exit state mapping Statements ist

Syntax

```
create [ or alter ] exit state mapping mappingname
with map = ( statename { , signed_integer , statename } )
```

Beschreibung

Das create exit state mapping Statement definiert das Mapping von Exit Codes zu logischen Exit States. Die einfachste Form des Statements spezifiziert nur einen Exit State. Das bedeutet, dass der Job automatisch diesen Exit State nach Ablauf bekommt, ungeachtet seines Exit Codes. Komplexere Definitionen spezifizieren mehr als einen Exit State und mindestens eine Abgrenzung.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Beispiel

Das untenstehende Beispiel zeigt ein relativ einfaches jedoch realistisches Map-Beispiel ping von Exit Codes nach logischen Exit States. Das Statement

```
create exit state mapping beispiel1
with map = ( fehler,
    0, erfolg,
    1, warnung,
    4, fehler);
```

definiert folgendes Mapping:

Exit code	Exit code	Resultierende
Range von	Range bis	exit state
$-\infty$	-1	fehler
0	0	erfolg
1	3	warnung
4	∞	fehler

create exit state profile

create exit state profile

Zweck

Zweck

Das create exit state profile Statement wird eingesetzt um eine Reihe von gültigen Exit States zu definieren.

Syntax

Syntax

Die Syntax des create exit state profile Statements ist

```
create [ or alter ] exit state profile profilename
with WITHITEM {, WITHITEM}

WITHITEM:
    default mapping = < none | mappingname >
    | force
    | state = ( ESP_STATE {, ESP_STATE} )

ESP_STATE:
    statename < final | restartable | pending > [ OPTION { OPTION} ]

OPTION:
    < unreachable | broken | batch default | dependency default >
```

Beschreibung

Beschreibung

Das create exit state profile Statement wird benutzt um eine Menge von gültigen Exit States für einen Job, Milestone oder Batch zu definieren.

default mapping Mit der **default mapping** Klausel ist es möglich zu definieren welches Exit State Mapping benutzt werden soll, wenn kein anderes Mapping spezifiziert ist. Dieses vereinfacht die Erstellung von Jobs wesentlich.

force Während ein Exit State Profile erstellt wird, hat die force Option keinen Effekt und wird ignoriert. Wenn "or alter" spezifiziert ist und der Exit State Profile, der erstellt werden soll, bereits existiert, verschiebt die force Option die Integritätsprüfung auf später.

state Die **state** Klausel definiert welche Exit State Definitions innerhalb dieses Profiles gültig sind. Jede Exit State Definition muss als **final**, **restartable** oder **pending** klassifiziert sein. Wenn ein Job ein State der **final** ist erreicht, kann dieser Job nicht mehr gestartet werden, das bedeutet, der State kann sich nicht mehr ändern. Wenn ein Job ein State der **restartable** ist erreicht, kann er noch einmal gestartet werden. Das bedeutet, dass sich der State von solch einem Job ändern kann. **Pending** bedeutet, das ein Job nicht neu gestartet werden kann, aber auch nicht final ist. Der Status muss von außerhalb gesetzt werden.

Die Reihenfolge, in der die Exit States definiert sind, ist relevant. Der erste spezifizierte Exit State hat die höchste und der zuletzt spezifizierte Exit State hat die niedrigste Präferenz. Normalerweise werden **final** States später als **restartable** States spezifiziert. Die Präferenz eines States wird benutzt um zu entscheiden welcher State sichtbar ist, wenn mehrere verschiedene Exit States von Children zusammengeführt werden.

Man kann genau ein Exit State als **unreachable** State deklarieren. Das bedeutet, ein Job, Batch oder Milestone mit diesem Profile bekommt den spezifizierten State, sobald er unreachable geworden ist. Dieses Exit State muss **final** sein.

Maximal ein Exit State innerhalb eines Profiles kann als **broken** State gekennzeichnet werden. Das bedeutet, ein Job wird diesen State erreichen, so bald der Job in den **error** oder **broken_finished** State gewechselt ist. Dieses kann mittels eines Triggers behandelt werden. Der Exit State der als ein **broken** State definiert ist, muss **restartable** sein.

Maximal ein State kann als **batch default** deklariert werden. Ein leerer Batch wird diesen Status annehmen. Damit kann explizit vom standard Verhalten abgewichen werden. Wird kein Status als **batch default** gekennzeichnet, wird ein leerer Batch automatisch den, nicht als **unreachable** gekennzeichnete, finalen Status mit niedrigster Präferenz annehmen. Gibt es einen solchen Status nicht, wird auch der **unreachable** State als Kandidat betrachtet.

Beliebig viele finals States können als **dependency default** gekennzeichnet werden. Dependencies die eine default Abhängigkeit definieren werden dann erfüllt, wenn der required Job eine der States, die als **dependency default** gekennzeichnet sind, annimmt.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Beispiel

In diesen Beispielen werden die Exit State Profiles beispiel_1 und beispiel_2 Beispiel erstellt.

Im ersten, sehr einfachen Beispiel, soll der Exit Status erfolg ein final Status sein.

create exit state profile beispiel_1

create exit state profile

```
with
    state = ( erfolg final
);
```

Im zweiten Beispiel wird der Exit Status fehler als restartable definiert. Dieser Status hat eine höhere Priorität als der (final) Status success und muss dementsprechend als erste aufgeführt werden.

create folder

User Commands

create folder

Zweck

Das create folder Statement wird eingesetzt um eine Mappe für Job Definitionen Zweck und/oder für andere Folder zu erstellen.

Syntax

Die Syntax des create folder Statements ist

Syntax

```
create [ or alter ] folder folderpath [ with WITHITEM {, WITHITEM} ]
WITHITEM:
     environment = < none | environmentname >
     group = groupname [ cascade ]
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     parameter = none
     parameter = ( parametername = string {, parametername = string} )
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
```

Beschreibung

use view

Dieses Kommando erstellt einen Folder und kennt folgende Optionen:

Beschreibung

environment option Wenn ein Environment einem Folder zugewiesen wurde, wird jeder Job in dem Folder und seinen Subfolders alle Resource Anforderungen, von der Environment Definition, erben.

create folder

group option Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

parameter option Mit der parameter Option ist es möglich key/value pairs für den Folder zu definieren. Die vollständige Liste der Parameter muss innerhalb eines Kommandos spezifiziert werden.

inherit grant option Die **inherit grants** Klausel ermöglicht es zu definieren welche Privilegien über die Hierarchie geerbt werden sollen. Wird dieser Klausel nicht spezifiziert, werden per Default alle Rechte geerbt.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create footprint

User Commands

create footprint

Zweck

Das create footprint Statement wird eingesetzt um eine Anzahl von häufig be- Zweck nutzten System Resource Anforderungen zu erstellen.

Syntax

Die Syntax des create footprint Statements ist

Syntax

```
create [ or alter ] footprint footprintname
with resource = ( REQUIREMENT {, REQUIREMENT} )

REQUIREMENT:
ITEM { ITEM}

ITEM:
    amount = integer
    | < nokeep | keep | keep final >
    | resourcepath
```

Beschreibung

Das create footprint Kommando erstellt eine Menge von Resource-Anforderungen Beschreibung welche wiederbenutzt werden können. Die Required Resources sind alle System Resources. Die Required Resources werden durch ihren Namen beschrieben, eine Menge, per Default Null und optional eine keep Option.

keep Die keep Option in einer Resource-Anforderung definiert den Zeitpunkt zu dem die Resource freigegeben wird. Die keep Option ist sowohl für System als auch für Synchronizing Resources gültig. Es gibt drei mögliche Werte. Ihre Bedeutung wird in der folgenden Tabelle erklärt:

Wert	Bedeutung	
nokeep	Die Resource wird am Jobende freigegeben. Dies ist	
	das Default-Verhalten	
keep	Die Resource wird so bald der Job den Final State er-	
	reicht hat freigegeben	
keep final	Die Resource wird freigegeben, wenn der Job und alle	
	seine Children final sind	

create footprint

amount Die amount Option ist nur mit Anforderungen für Named Resources von der Art "System" oder "Synchronizing" gültig. Der Amount in einer Resource-Anforderung drückt aus, wieviele Einheiten von der Required Resource belegt werden.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create group

User Commands

create group

Zweck

Das create group Statement wird eingesetzt um ein Objekt zu erstellen, an dem Zweck man Rechte vergeben kann.

Syntax

Die Syntax des create group Statements ist

Syntax

```
create [ or alter ] group groupname [ with WITHITEM ]

WITHITEM:
    user = none
    | user = ( username {, username} )
```

Beschreibung

Das create group Statement wird benutzt um eine Gruppe zu erstellen. Ist das Schlüsselwort "or alter" spezifiziert, wird eine bereits existierende Gruppe geändert. Ansonsten wird eine bereits existierende Gruppe als Fehler betrachtet.

Beschreibung

user Die **user** Klausel wird benutzt um zu spezifizieren, welche Benutzer Gruppenmitglieder sind.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

create interval

create interval

Zweck

Zweck

Das create interval Statement wird eingesetzt um ein periodisches oder nicht periodisches Muster, auf welchem Events getriggert werden können, zu definieren.

Syntax

Syntax

Die Syntax des create interval Statements ist

```
create [ or alter ] interval intervalname [ with WITHITEM {, WITHITEM} ]
WITHITEM:
     base = < none | period >
     duration = \langle \underline{none} \mid period \rangle
     embedded = < none | intervalname >
     endtime = < none | datetime >
     filter = none
     filter = ( intervalname {, intervalname} )
     < <u>noinverse</u> | inverse >
     selection = none
     selection = (IVAL_SELITEM {, IVAL_SELITEM})
     starttime = < none | datetime >
     synctime = datetime
     group = groupname
IVAL SELITEM:
< signed_integer | datetime | datetime - datetime >
```

Beschreibung

Beschreibung

Die Intervalle sind das Herzstück des Time Schedulings. Sie können als Muster von Blöcken gesehen werden. Diese Muster können periodisch oder aber auch aperiodisch sein. Innerhalb einer **Periode** (**Base**), die im aperiodischen Fall eine Länge unendlich (∞) hat, gibt es Blöcke einer bestimmten Länge (**Duration**). Der letzte Block kann dabei unvollständig sein falls die Periodenlänge kein ganzzahliges Vielfaches der Duration ist. Die Duration kann ebenfalls eine Länge ∞ haben. Das bedeutet, daß die Blöcke dieselbe Länge wie die Perioden haben.

Nun müssen nicht alle Blöcke auch tatsächlich vorhanden sein. Welche Blöcke vorhanden sind, kann gewählt werden. Dieses **Wählen** kann durch Angabe der Blocknummer relativ zum Anfang oder Ende einer Periode (1, 2, 3 bzw. -1, -2, -3) oder durch "von - bis" Angaben (alle Tage zwischen 3.4. und 7.6.) erfolgen.

98 I

create interval

User Commands

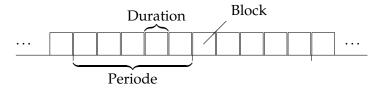


Abbildung 6.1.: Darstellung von Perioden und Blöcken

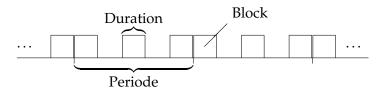


Abbildung 6.2.: Komplexeres Muster

Dadurch entstehen kompliziertere Muster wie in Abbildung 6.2.

Das Auswählen ist 1-based, d.h. der erste Block hat die Nummer 1. Der letzte Block wird mittels Nummer -1 addressiert. Ein Block 0 existiert somit nicht.

Ein Intervall kann im Wesentlichen mit folgenden Parametern beschrieben werden: Basisfrequenz (Periodenlänge), Duration und Auswahl. Da ein Intervall aber nicht zwingend immer gültig sein muss, kann weiterhin noch ein Start- und Endezeitpunkt angegeben werden.

Unendliche Intervalle Bei einem aperiodischen Intervall ohne Duration (Unendlichkeit) kommt dem Startzeitpunkt eine Sonderrolle zu: er definiert dann die einzige positive Flanke dieses Intervalls. Analog dazu legt ein Endzeitpunkt die einzige negative Flanke fest.

Wird eine Auswahl getroffen, führt diese Auswahl jeweils zu der Entstehung von Blöcken. Eine Auswahl "-0315T18:40" führt jedes Jahr am 15. März zu einem Block von 18h40 bis 18h41.

Es ist selbstverständlich, dass eine Auswahl von Blöcken über die Position (erster, zweiter, ..) natürlich unsinnig ist. Dies wird bei unendlichen Intervallen dann auch ignoriert.

Inverse Wenn etwa die Zeit zwischen Weihnachten und Neujahr positiv zu irgendeinem Zweck definiert wurde, gibt es bisher keine Möglichkeit die komplementierende Zeit leicht zu definieren. Im aktuellen Beispiel ist das noch nicht gravierend, bei komplexeren Muster führt diese Unmöglichkeit zu aufwendigen und fehlerträchtigen Doppeldefinitionen.

Es wird daher ein **Inverse**-Flag eingeführt, welches zur Folge hat, dass die angegebene Auswahlliste komplementär interpretiert wird, d.h. es werden nur die Blöcke ausgewählt, die ohne gesetztes Invert-Flag nicht ausgewählt würden. Im Falle des

create interval

Monatsultimo (letzter Arbeitstag des Monats) resultiert das Setzen des Inverse-Flags also in allen Arbeitstagen, mit Ausnahme des Monatsultimo.

Filter Der Auswahl von Blöcken kann noch weiter eingeschränkt werden. Hat man zum Beispiel einen Intervall "Tag des Monats" definiert (d.h. die Base ist einen Monat, die Duration einen Tag) und dann den zweiten Block ausgewählt, würde ein solcher Intervall jeweils am zweiten Tag eines Monats einen Block haben. Möchte man, dass dies nur für die ungerade Monate (Januar, März, Mai, ...) definieren, dann wäre dies ohne Filterfunktionalität wegen Schaltjahre nicht möglich.

Die Lösung des Problems besteht darin, dass ein weiterer Intervall (Monat des Jahres), mit der Selektion 1, 3, 5, 7, 9, 11 definiert wird ind dieser Intervall als Filter des ersten Intervalls angegeben wird.

Es gilt dann, dass der erste Intervall nur dann einen Block zeigt, wenn auch der zweite zu der "Zeit" einen Block zeigt.

Werden mehrere Intervalle als Filter angegeben, reicht es, wenn einer dieser Intervallen einen Block zum verlangten Zeitpunkt hat (ODER). Für die Abbildung einer AND-Beziehung zwischen den Filterintervallen, werden die Filterintervalle als Kette angelegt (A filtert B filtert C ... usw.). Die Reihenfolge der Filter ist dabei unwichtig.

Embedded Leider ist die Welt nicht immer ganz so einfach. Insbesondere ist es nicht unwichtig, ob man zuerst eine Operation ausführt, und dann seine Auswahl trifft, oder zuerst wählen muss und anschließend die Operation anwendet. Anders gesagt, ist es ein großer Unterschied, ob man über den letzten Tag des Monats – falls dies ein Arbeitstag ist – redet, oder über den letzten Arbeitstag des Monats. Diese Möglichkeit der Differenzierung wollen wir natürlich auch in unser Modell aufnehmen. Dazu wird die Möglichkeit des **Einbettens** eingeführt.

Beim Einbetten werden zuerst alle Parameter des eingebetteten Intervalls übernommen. Anschließend wird die Auswahlliste evaluiert. Obwohl erlaubt, hat dabei die Angabe einer "von - bis" Auswahl natürlich keinen Sinn, da diese Funktionalität mittels einer einfachen Multiplikation ebenfalls erzielbar ist. Viel interessanter ist die Möglichkeit der relativen Auswahl. Wenn etwa die Arbeitstage eines Monats eingebettet werden und anschließend der Tag -1 ausgewählt wird, haben wir insgesamt ein Intervall, dass den letzten Arbeitstag eines Monats definiert. Wird dagegen das Intervall mit den Arbeitstagen eines Monats mit einem Intervall, das den letzten Tag eines Monats liefert, multipliziert, erhalten wir nur dann einen Treffer, wenn der letzte Tag des Monats ein Arbeitstag ist.

Das Einbetten kann auch so verstanden werden, dass bei der Auswahl der Blöcke nicht *alle* eingebetteten Blöcke betrachet (und vor allem gezählt) werden, sondern statt dessen nur die *aktiven* Blöcke berücksichtigt werden.

Synchronisation Nicht berücksichtigt sind noch die Situationen, in denen Vielfache einzelner Perioden im Spiel sind. Eine Periode von z.B. 40 Tagen könnte ihre steigende

Flanke zu jeder beliebigen Mitternacht (00:00h) haben. Daher wird ein Synchronisationszeitpunkt (**synctime**) eingeführt, der die früheste Flanke auswählt, die \geq diesem Termin ist. Wird kein solcher Zeitpunkt explizit angegeben, wird das Datum der Definitionserstellung (*create*) verwendet.

Der erste Block einer Periode beginnt zunächst grundsätzlich an deren Beginn. In Fällen, in denen das nicht möglich ist (Periode = ∞ , Duration > Periode, Periode XOR Duration haben die Einheit Woche), wird der Beginn der Periode als Synchronisationszeitpunkt verwendet. Ist auch das nicht möglich (Periode = ∞), kommt der normale Synchronisationszeitpunkt zum Tragen. Dieses Vorgehen hat zur Folge, dass auch der *erste* Block einer Periode unvollständig sein kann (und dann *niemals* aktiv ist).

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

I 101

create job definition

create job definition

Zweck

Zweck

Das create job definition Statement wird eingesetzt um ein Scheduling Entity Objekt zu erstellen welches selbstständig oder als Teil einer grösseren Hierarchie submitted werden kann.

Syntax

Syntax Die Syntax des create job definition Statements ist

resource = none

runtime = integer
runtime final = integer

```
create [ or alter ] job definition folderpath . jobname
with WITHITEM {, WITHITEM}
WITHITEM:
     children = none
     children = ( JOB_CHILDDEF {, JOB_CHILDDEF} )
     dependency mode = < all \mid any >
     environment = environmentname
     errlog = < <u>none</u> | filespec [ < notrunc | trunc > ] >
     footprint = < none | footprintname >
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     kill program = < none | string >
     logfile = < <u>none</u> | filespec [ < notrunc | trunc > ] >
     mapping = < none | mappingname >
     < <u>nomaster</u> | master >
     nicevalue = < none | signed_integer >
     parameter = none
     parameter = ( JOB_PARAMETER {, JOB_PARAMETER} )
     priority = < none | signed_integer >
     profile = profilename
     required = none
     required = ( JOB_REQUIRED {, JOB_REQUIRED} )
     rerun program = < none | string >
```

resource = (REQUIREMENT {, REQUIREMENT})

run program = < none | string >

< nosuspend | suspend >

< <u>noresume</u> | resume in period | resume at datetime >

```
timeout = none
     timeout = period state statename
     type = < job | milestone | batch >
     group = groupname
     workdir = < none | string >
JOB_CHILDDEF:
JCD_ITEM { JCD_ITEM}
PRIVILEGE:
     create content
     drop
     edit
     execute
     monitor
     operate
     resource
     submit
     use
     view
JOB_PARAMETER:
parametername < [ JP_WITHITEM ] [ default = string ] | JP_NONDEFWITH >
[local] [ < export = parametername | export = none > ]
JOB_REQUIRED:
JRQ_ITEM { JRQ_ITEM}
REQUIREMENT:
JRD_ITEM { JRD_ITEM}
JCD_ITEM:
     alias = < \underline{none} \mid alias name >
     < enable | disable >
     folderpath.jobname
    ignore dependency = none
    ignore dependency = ( dependencyname {, dependencyname} )
     < childsuspend | suspend | nosuspend >
    merge mode = < nomerge | merge local | merge global | failure >
  nicevalue = < <u>none</u> | <u>signed_integer</u> >
```

```
User Commands
```

create job definition

```
priority = < none | signed_integer >
   < noresume | resume in period | resume at datetime >
   < static | dynamic >
JP_WITHITEM:
     import
   parameter
     reference child folderpath (parametername)
     reference folderpath ( parametername )
     reference resource resourcepath (parametername)
JP_NONDEFWITH:
     constant = string
  JP_AGGFUNCTION ( parametername )
JRQ_ITEM:
     dependency dependencyname
     folderpath.jobname
     mode = \langle \underline{all final} \mid job final \rangle
     state = none
     state = ( JRQ_REQ_STATE {, JRQ_REQ_STATE} )
     state = all reachable
     state = default
     state = unreachable
     unresolved = < error | ignore | suspend | defer >
JRD_ITEM:
     amount = integer
     expired = < none | signed_period >
     < nokeep | keep | keep final >
     lockmode = LOCKMODE
     nosticky
     resourcepath
     state = none
     state = ( statename {, statename} )
     state mapping = < none | rsmname >
     sticky
  [ ( < identifier | folderpath | identifier , folderpath | folderpath , identifier > ) ]
```

```
JP_AGGFUNCTION:
    avg
    | count
    | max
    | min
    | sum

JRQ_REQ_STATE:
    statename [ < condition = string | condition = none > ]

LOCKMODE:
    n
    | s
    | sc
    | sx
    | x
```

Beschreibung

Dieses Kommando erzeugt oder wahlweise ändert Job, Batch oder Milestone Beschreibung Definitions.

Da Jobs, Batches und Milestones eine Menge gemeinsam haben, benutzen wir im Folgenden meistens den generellen Fachausdruck "Scheduling Entity" wann immer das Verhalten für alle drei Typen von Job Definitions das gleiche ist. Die Ausdrücke "Job", "Batch" und "Milestone" werden für Scheduling Entities des entsprechenden Typs Job, Batch und Milestone benutzt. Wenn der "or alter" Modifikator benutzt wird, wird das Kommando, falls bereits ein Scheduling Entity mit dem gleichen Namen existiert, dieses entsprechend den spezifizierten Optionen ändern.

aging Das aging beschreibt die Geschwindigkeit mit der die Priorität hochgestuft wird.

children Der Children Abschnitt eines Job Definition Statements definiert eine Liste von Child Objekten und wird benutzt um eine Hierarchie aufzubauen, die das Modellieren von komplexen Job Strukturen ermöglicht.

Wann immer ein Scheduling Entity submitted wird, werden alle statischen Children rekursiv submitted.

Zusätzlich können Children, die nicht statisch sind, während der Ausführung, durch einen Running Job oder Trigger, submitted werden.

Die Children werden durch eine Komma getrennte Liste von Scheduling Entity Pfadnamen und zusätzliche Eigenschaften spezifiziert.

Die Eigenschaften der Child Definitions sind im folgenden Beschrieben:

<u>ALIAS</u> Mittels dieser Option kann die Implementierung des submitteten Jobs unabhängig von der Folder-Struktur gehalten werden und es wird unabhängig davon, ob Objekte innerhalb der Folder-Struktur verschoben werden, funktionieren.

Die alias einer child Definition wird nur benutzt wenn Jobs dynamische Children submitten.

IGNORE DEPENDENCY Normalerweise werden Abhängigkeiten der Parents auf die Children vererbt. In wenige besondere Situationen kann dies jedoch unerwünscht sein. Mit der **ignore dependency** Option können solche Abhängigkeiten daher ignoriert werden.

MERGE MODE Ein einzelnes Scheduling Entity kann als Child von mehr als ein Parent Scheduling Entity benutzt werden. Wenn zwei oder mehr solcher Eltern ein Teil eines Master Runs sind, werden die selben Children mehrmals innerhalb dieses Master Runs instanziiert. Das ist nicht immer eine gewünschte Situation. Das Setzen des Merge Modes kontrolliert wie das System damit umgeht.

Die folgende Tabelle gibt einen Überblick über die möglichen Merge Modes und ihrer Bedeutung:

merge mode	Description
nomerge	Eine doppelte Instanz von dem Scheduling Entity wird erstellt. Das ist das Default Verhalten.
merge global	Es wird keine doppelte Instanz erstellt. Ein Link wird zwischen dem Parent Submitted Entity und dem be- reits existierendem Child Submitted Entity erstellt.
merge local	Wie Merge Global, aber nur Submitted Entities, die in einem einzelnen Submit erzeugt wurden, werden zusammengeführt.
failure	Der Submit, der versucht einen doppelten Submitted Entity zu erstellen schlägt fehl.

NICEVALUE Die nicevalue definiert ein Offset der Priorität, der für die Berechnung der Prioritäten des Childs und seine Children. Werte von -100 bis 100 sind zulässig. PRIORITY Die spezifizierte Priorität in einer Child Definition überschreibt die Priorität von der Child Scheduling Entity Definition. Werte von 0 (hohe Priorität) bis 100 (niedrige Priorität) sind zugelassen.

TRANSLATION Das Setzen der Exit State Translation für ein Child, resultiert in einer Übersetzung der Exit State des Childs zu einer Exit State welche in der resultierenden Exit State von dem Parent Submitted Entity merged wird.

Wenn keine Übersetzung gegeben ist, wird ein Child State, der nicht gleichzeitig ein gültiger Parent State ist, ignoriert.

Wenn eine Übersetzung gegeben ist, müssen alle Child States nach einem gültigen Parent State übersetzt werden.

<u>SUSPEND CLAUSE</u> Die child suspend Klausel definiert ob ein, in dem Kontext dieser Child Definition, neuer Submitted Job suspended wird.

Die folgende Tabelle zeigt die möglichen Werte und deren Bedeutung von der suspend Klausel an:

suspend clause	Description
suspend	Das Child wird unabhängig von dem Wert des suspend flags, spezifiziert in der Child Scheduling Entities, suspended.
nosuspend	Das Child wird unabhängig von dem Wert des suspend flags, spezifiziert in der Child Scheduling Entities Definition, nicht suspended.
childsuspend	Das Child wird suspended, wenn das suspended Flag in dem Child Scheduling Entity gesetzt ist.

Falls **suspend** spezifiziert wird, kann wahlweise auch eine Resume Klausel mitgegeben werden, die ein automatisches Resume zum angegebenen Zeitpunkt, bzw. nach Ablauf des angegebenen Intervalls zur Folge hat.

Für teilqualifizierte Zeitpunkten wird die Submitzeit als Referenz genommen. So bedeutet etwa T16:00, dass der Job bei einer Submitzeit von 15:00 nach etwa einer Stunde anläuft. Liegt der submitzeitpunkt jedoch nach 16:00, wird der Job bis zum nächsten Tag warten.

<u>DYNAMIC CLAUSE</u> Die child dynamic Klausel definiert ob das Child immer dann von dem System automatisch submitted wird, wenn auch der Parent submitted wird.

Dynamische Children werden in dem Kontext Trigger Definitionen und programmatische Submits von Running Jobs benutzt. Um imstande zu sein ein Child zu submitten, muss dieses Child als ein dynamisches Child definiert sein.

Die folgende Tabelle zeigt die möglichen Werte in der dynamic Klausel und ihre Bedeutung an.

dynamic clause	Description
static	Das Child wird automatisch mit dem Parent submit-
	ted.
dynamic	Das Child wird nicht automatisch mit dem Parent submitted.

Milestones haben eine andere Semantik für ihre Children. Wann immer ein Scheduling Entity in einem Master Run dynamisch submitted wird, welches auch ein Child von einem Milestone in dem selben Master Run ist, wird der Submitted Scheduling Entity als Child an diesem Milestone gebunden. Somit kann ein Milestone nur dann Final werden, wenn seine Abhängigkeiten erfüllt und alle Children final sind.

In anderen Worten, ein Milestone sammelt Child-Instanzen die bei anderen Submitted Entities dynamisch submitted werden und wartet auf die Beendigung von diesen Submitted Entities. Um dies korrekt funktionieren zu lassen, sollte eine Abhängigkeit, von dem Scheduling Entity der submitted, definiert sein.

dependency mode Der dependency mode definiert welche Required Submitted Entities einen Final State erreichen müssen, bevor der abhängige Submitted Entity den 'Dependency Wait' System State verlassen kann.

Die folgende Tabelle zeigt die möglichen Dependency Modes und ihre Bedeutung.

dependency mode	Beschreibung	
all	Der Submitted Entity verlässt den Dependency Wait	
	State, nachdem alle Abhängigkeiten erfüllt sind.	
any	Der Submitted Entity verlässt den Dependency Wait	
	State, nachdem mindestens eine Abhängigkeit erfüllt	
	ist.	

environment Jeder Job muss definieren welches Environment für die Jobausführung gebraucht wird.

Der Job kann nur von Jobserver ausgeführt werden welche alle Static Resource Anforderungen die in der Environment Definition gelistet sind erfüllen. Die environment Option ist nur für Jobs gültig.

errlog Die errlog Option definiert die Datei in die Fehlerausgaben (stderr) des auszuführenden Prozesses geschrieben werden. Wenn der Dateiname relativ ist, wird die Datei relativ zu dem working Directory des Jobs angelegt. Diese Option ist nur für Jobs gültig.

footprint Footprints sind Mengen von Anforderungen für System Resources. Wenn mehrere Jobs mit ähnlichen Anforderungen definiert werden, wird das durch die Benutzung von Footprints viel einfacher.

Der Job kann nur von Jobserver ausgeführt werden, die alle System Resource Anforderungen erfüllen, die in der Footprint Definition gelistet sind. Die Footprint Option gilt nur für Jobs.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

inherit grant Die **inherit grants** Klausel ermöglicht es zu definieren welche Privilegien über die Hierarchie geerbt werden sollen. Wird dieser Klausel nicht spezifiziert, werden per Default alle Rechte geerbt.

kill program Diese Option wird benutzt um die Möglichkeit zu schaffen, laufende Prozesse aus dem Scheduling System heraus, vorzeitig terminieren zu können.

Üblicherweise beinhaltet das Kill-Programm die PID des Running Jobs als Parameter (z.B. kill -9 \${PID}).

Für Details über Kommandozeilen parsing, Ausführungen und Parameter Substitutionen siehe die "run program" Option auf Seite 118.

logfile Die logfile Option definiert die Datei in die der standard Output (stdout) des auszuführenden Prozesses geschrieben wird. Wenn der Dateiname relativ ist, wird die Datei relativ zu dem working Directory des Jobs angelegt. Diese Option ist nur für Jobs gültig.

mapping Die mapping Option definiert das exit state mapping das benutzt wird um Betriebssystem Exit Codes eines ausführenden Programms in einen Exit State zu übersetzen. Wenn ein Job kein mapping hat, wird der default Exit State Mapping des Exit State Profiles, des Jobs, benutzt.

Für eine ausführliche Beschreibung des exit state mappings siehe das "create exit state mapping" Kommando auf Seite 89.

nicevalue Die nicevalue Option definiert eine Korrektur die für die Berechnung der Prioritäten des Jobs und seinen Children benutzt wird. Es sind Werte von -100 bis 100 erlaubt.

parameter Die parameter Section definiert welche Parameter und Inputwerte ein Job benötigt und wie der Job Daten mit anderen Jobs und dem Scheduling System auswechselt.

Die Parameter können in der Spezifikation des Run Programms, Rerun Programms, Kill Programms, Workdir, Logfile und error Logfile, sowie in Trigger und Dependency Conditions genutzt werden.

Zusätzlich kann ein Job zur Laufzeit Parameter abfragen oder setzen. Variablen die zur Laufzeit gesetzt und nicht von der Job Definition definiert wurden, sind nur für den Job selbst sichtbar und können nicht referenziert werden. Dasselbe gilt selbstverständlich ebenfalls für alle Variablen die als **local** definiert sind, sowie für die unten genannte Systemvariablen.

Gelegentlich gibt es jedoch die Notwendigkeit eine oder mehrere der (z.B.) Systemvariablen nach außen bekannt zu machen. Dies kann mittels eines kleinen Tricks einfach gemacht werden. Enthählt der Wert eines Parameters eine Zeichenkette der Form \$irgendwas (also ein \$-Zeichen gefolgt von einem Namen), wird dies als der Name einer Variable interpretiert und es wird versuch diese Variable *im Scope des Objektes der den ursprünglichen Wert des Parameters geliefert hat* auf zu lösen.

create job definition

So definiert z.B. ein Job System. A eine Konstante namens Myjobname mit als Inhalt \$jobname. Wird nun etwa über eine Reference die Konstante Myjobname von aussen angesprochen, bekommt man als Ergebnis den Wert System. A.

Es sind für jeden Job immer eine Anzahl von System Variablen definiert. Diese werden vom System gesetzt und stehen dem Job für einen lesenden Zugriff zur Verfügung.

Diese System Variablen sind:

Name	Description		
JOBID	Submitted entity id des Jobs		
MASTERID	Submitted entity id des Master Jobs oder Batches		
KEY	"Passwort" des Jobs für die Verbindung zum		
	Scheduling System als Job mit "JOBID"		
PID	Die Betriebssystem Prozess id des Jobs. Dieser Parameter ist nur bei Kill Programs gesetzt		
LOGFILE	Name des Logfiles (stdout)		
ERRORLOG	Name des Error Logfiles (stderr)		
SDMSHOST	Hostname des Scheduling Servers		
SDMSPORT	Listen Port des Scheduling Servers		
JOBNAME	Name des Jobs		
JOBTAG	Childtag des Jobs ist gegeben wenn der Job dy- namisch Submitted wird		
TRIGGERNAME	Name des Triggers		
TRIGGERTYPE	Typ des Triggers (JOB_DEFINITION oder NAMED_RESOURCE)		
TRIGGERBASE	Name des triggernden Objektes der den Trigger zum feuern veranlasst		
TRIGGERBASEID	Id der triggernden Objekt Definition die den Trigger zum feuern veranlasst		
TRIGGERBASEJOBID	Id des triggernden Objektes der den Trigger zum feuern veranlasst		
TRIGGERORIGIN	Name des triggernden Objektes welches den Trigger definiert		
TRIGGERORIGINID	Id der triggernden Objekt Definition welche den Trigger definiert		
TRIGGERORIGINJOBID	Id des triggernden Objektes welches den Trigger definiert		
TRIGGERREASON	Name des triggernden Objektes welches den Trigger direkt oder indirekt feuert		
TRIGGERREASONID	Id der triggernden Objekt Definition die den Trigger direkt oder indirekt feuert		
Continued on next page			

Continued from previous page	
Name	Description
TRIGGERREASONJOBID	Id des triggernden Objektes welches den Trigger di-
	rekt oder indirekt feuert
TRIGGERSEQNO	Die Anzahl mal die der Trigger feuerte
TRIGGEROLDSTATE	Der alte Status des Objektes, verursacht durch den Trigger für Resource Trigger
TRIGGERNEWSTATE	(Neuer) Status des Objektes welches bewirkt das der Trigger ausgelöst wird
SUBMITTIME	Submit-Zeitpunkt
STARTTIME	Start Zeitpunkt
EXPRUNTIME	Erwartete Laufzeit
JOBSTATE	Exit State des Jobs
MERGEDSTATE	Merged Exit State des Jobs
PARENTID	ID des Parent Jobs (Submission Baum)
STATE	Aktueller Status des Jobs (Running, Finished,)
ISRESTARTABLE	Ist der Job Restartable? $1 = ja$, $0 = nein$
SYNCTIME	Zeitpunkt des Übergangs nach Synchronize Wait
RESOURCETIME	Zeitpunkt des Übergangs nach Resource Wait
RUNNABLETIME	Zeitpunkt des Übergangs nach Runnable
FINISHTIME	Abschluss Zeitpunkt
SYSDATE	Aktuelles Datum
SEID	Id der Job Definition
TRIGGERWARNING	Der Text der Warnung die diesen Trigger ausgelöst hat
LAST_WARNING	Der Text der zuletzt ausgegebenen Warnung. Wenn keine aktuelle Warnung vorliegt ist er leer
RERUNSEQ	Die Anzahl der Reruns bis jetzt
SCOPENAME	Name des Scopes (Jobservers) in dem der Job läuft oder zuletzt lief

Tabelle 6.1.: List of System Variables

Die TRIGGER... System Variablen sind nur gefüllt, wenn der Job von einem Trigger submitted wurde. Für eine ausführlichere Beschreibung der TRIGGER... System Variablen, siehe das create trigger Statement auf Seite 138.

Wenn ein Job ausgeführt wird, werden die Parameter die in Kommandos, workdir und Datei Spezifikationen benutzt werden aufgelöst, konform unterstehender Reihenfolge:

- 1. System Variable
- 2. Der Jobeigene Adressraum
- 3. Der Adressraum des Jobs und der submittenden Parents, von unten nach oben

create job definition

- 4. Der Adressraum des Jobservers der den Job ausführt
- 5. Der Adressraum der Parent Scopes des Jobservers, der den Job ausführt, von unten nach oben
- 6. Die Job Definitions Parent Folders, von unten nach oben
- 7. Die Parent Folders des Parent Jobs, von unten nach oben

Wenn der Konfigurations Parameter 'ParameterHandling' des Servers auf 'strict' (default) gesetzt ist, führt der Zugriff auf Variablen die in der Job Definition nicht definiert sind zu einer Fehlermeldung, es sei denn es handelt sich um eine Systemvariable.

Wenn im Inhalt einer Variable eine Referenz auf einen weiteren Parameter auftritt, wird dieser Parameter im Kontext des definierenden Jobs ausgewertet und ersetzt. Die verschiedenen Parametertypen und ihre Semantik werden im Folgendem beschrieben:

IMPORT Import type Parameter werden benutzt um die Daten eines Job Scheduling Environments einem anderen Job zu übergeben. Dieser Type ist beinahe wie der type Parameter, doch import type Parameter können nicht wie Parameter übergeben werden, wenn ein Job submitted wird. Import type Parameter können einen Default-Wert haben, welcher benutzt wird wenn kein Wert von dem Scheduling Environment erworben werden kann.

PARAMETER Parameter vom type Parameter werden benutzt um Daten von einem Job Scheduling Environment an einem anderen Job zu übergeben. Dieser Typ ist fast wie der type import, doch parameter type Parameter können als Parameter übergeben werden wenn ein Job submitted wird. Parameter type Parameter können einen Default Wert haben, welcher benutzt wird wenn kein Wert von dem Scheduling Environment erworben werden kann.

<u>REFERENCE</u> Referenz type Parameter werden normalerweise benutzt um Ergebnisse von einem Job zu einem anderen zu übergeben.

Zum Anlegen einer Referenz werden der vollqualifizierte Name der Job Definition, sowie der Name des referenzierenden Parameters benötigt. Beim Auflösen der Referenz wird das nächstliegende Submitted Entity gesucht, was der Job Definition der Referenz entspricht. Wenn diese Zuordnung nicht eindeutig gemacht werden kann, wird ein Fehler ausgegeben. Wird kein entsprechendes Submitted Entity gefunden, wird, soweit definiert, der Default-Wert zurückgegeben.

REFERENCE CHILD Child Reference Parameter werden genutzt um auf Parameter von direkten oder indirekten Children zu verweisen. Dies kann etwa zu reporting Zwecken nützlich sein. Die Definition eines Child Reference Parameters erfolgt mittels eines vollqualifizierten Job Definition Namens, sowie des Namens des zu qualifizierenden Parameters. Bei der Auflösung des Parameters wird in der Submission Hierarchie nach unten gesucht, anstelle nach oben wie bei den Reference Parameters. Das Verhalten bei der Auflösung ist ansonsten identisch mit der Auflösung vom Reference Parameter.

<u>REFERENCE RESOURCE</u> Resource reference type Parameter werden bezugnehmend auf Parameter von allokierten Resourcen benutzt.

Dieser Parameter Typ benötigt einen vollqualifizierten Namen einer Named Resource mit einem additionalen Parameternamen um die Referenzvorgabe zu spezifizieren. Voraussetzung für die Nutzung eines Resource Reference Parameters ist, dass die Resource auch angefordert wird. Die Wertermittlung erfolgt im Kontext der allokierten Resource.

<u>RESULT</u> Parameter vom Typ "result" können vom Job (mittels des API) einen Wert bekommen. Solange dieser Wert nicht gesetzt wurde, wird bei der Wertabfrage der optionale Default-Wert zurückgegeben.

<u>CONSTANT</u> Parameter vom Typ "Constant" sind Parameter mit einem bei der Definition festgelegten Wert. Dieser Wert kann sich also zur Laufzeit nicht ändern.

LOCAL Diese Variablen sind nur aus der Sicht des definierenden Jobs sichtbar.

priority Die priorität eines Jobs bestimmt die Ausführungsreihenfolge von Jobs. Werte von 0 (hohe Priorität) bis 100 (niedrige Priorität) sind zugelassen. Die priority Option gilt nur für Jobs.

profile Das profile definiert den Exit State Profile der die gültige Exit States des Scheduling Entity beschreibt.

Für eine ausführliche Beschreibung der Exit State Profiles siehe das "create exit state profile" Kommando auf Seite 90.

required Die 'required' section definiert die Abhängigkeiten von anderen submitteten Entities in einem Master Run, die erfüllt sein müssen bis das Submitted Entity fähig ist zu weiterzulaufen.

Ob alle Abhängigkeiten erfüllt sein müssen oder nur eine, wird durch den 'dependency mode' definiert.

Abhängigkeiten werden in einer kommagetrennten Liste von vollqualifizierten Namen von Scheduling Entities (inklusive Pfandnamen von Folder) definiert.

Abhängigkeiten wirken nur zwischen Submitted Entities des Master Runs. Die Synchronizierung der Submitted Entities aus unterschiedlichen Master Runs muss mittels Synchronizing Resources implementiert werden.

Nach dem Anlegen der Submitted Entity Instanzen der Submitted Scheduling Entity Hierarchie, sucht das System die Abhängigkeiten wie folgt: Beginnend bei dem Parent des Abhängigen Submitted Entity, wird in allen Children eine Instanz des Required Scheduling Entity gesucht, dabei wird natürlich der Ast mit dem abhängigen Submitted Entity ausgelassen. Wenn keine Instanz gefunden wird, geht die Suche bei den Submit Hierarchy Parents weiter, bis genau eine Instanz gefunden wird. Wenn keine Instanz gefunden wird, definiert die Eigenschaft 'unresolved' wie diese Situation von dem System gehandhabt wird. Wird mehr als ein

Submitted Entity gefunden, schlägt der Submit mit einem 'ambigous dependency resolution' Fehler fehl.

Während der Ausführung eines Master Runs, kann ein Scheduling Entity einen 'unreachable' State bekommen, da die Abhängigkeiten nicht mehr erfüllt werden können. Dies kann passieren, wenn ein Required Scheduling Entity einen Final State erreicht, der nicht in der Liste von benötigten States für Dependencies eingetragen ist oder durch das Canceln eines Submitted Entities der von einem anderen Submitted Entity benötigt wird. Diese zwei Fälle werden unterschiedlich gehandhabt

Wenn die unreachable Situation durch ein Submitted Entity verursacht wird, der mit einem nicht passenden Exit State beendet wird, ermittelt das System den Exit State Profile des abhängigen Submitted Entities und setzt den Exit State in den State der als 'unreachable' in dem Profile markiert ist.

Wenn keiner der Profile States als unreachable State markiert ist oder der unreachable Zustand durch das Canceln eines Submitted Entity verursacht wurde, wird das abhängige Submitted Entity in den Zustand unreachable versetzt, welcher nur von einem Operator dadurch aufgelöst werden kann, dass er die Abhängigkeit ignoriert oder das abhängige Entity cancelt.

Alle direkten oder indirekten Children eines Jobs oder Batches erben alle Abhängigkeiten des Parents. Das heißt, das kein Child eines Jobs oder Batches den Zustand dependency wait verlassen kann solange der Parent selber in den Zustand dependency wait ist. Children von Milestones erben die Abhängigkeiten vom Parent nicht.

Die Eigenschaften der dependency definitions werden im Folgenden beschrieben.

<u>CONDITION</u> Es ist möglich zu einer Dependency eine Condition an zu geben. Die Dependency ist erst dann erfüllt, wenn die Evaluation der Condition den Wahrheitswert "true" ergibt. Wird keine Condition spezifiziert, gilt die Bedingung immer als erfüllt

<u>DEPENDENCY NAME</u> Optional kann beim Definieren einer Abhängigkeit ein Name für die Abhängigkeit spezifiziert werden. Children, sowohl direkte als auch indirekte, können auf dem Namen Bezug nehmen um diese Abhängigkeit zu ignorieren.

MODE Die mode Eigenschaft ist nur relevant, wenn das benötigte Scheduling Entity ein Job mit Children ist. In diesem Fall definiert das Dependency Mode den Zeitpunkt in dem die Abhängigkeit erfüllt wird.

Die folgende Tabelle zeigt die zwei möglichen Werte und ihre Bedeutung:

dependency mode	Beschreibung
all_final	Der benötigte Job und alle seine Children müssen einen Final State erreicht haben
job_final	Nur der benötigte Job selbst muss einen Final State erreichen, der Zustand der Children ist irrelevant

<u>STATE</u> Die state Eigenschaft einer Abhängigkeit definiert eine Liste von Final States die das benötigte Scheduling Entity erreichen kann um die Abhängigkeit zu erfüllen.

Ohne diese Option ist die Abhängigkeit erfüllt, wenn das benötigte Scheduling Entity einen Final State erreicht.

Zusätzlich ist es möglich bei einem State eine Condition an zu geben. Wenn eine Condition spezifiziert wurde, gilt eine Abhängigkeit nur dann als erfüllt, wenn die Condition erfüllt ist. Die syntaktische Regeln für die Spezifikation von Conditions sind dieselbe die auch für Trigger gelten. Siehe dazu das create trigger Statement auf Seite 138.

Als weitere Möglichkeiten stehen einige implizite Definitionen zur Verfügung:

- default Die Dependency ist erfüllt wenn der Vorgänger einer der States, die in seinem Profile als dependency default gekennzeichnet sind, erreicht hat.
- all reachable Die Dependency ist erfüllt wenn der Vorgänger einer der States, die nicht als unreachable gekennzeichnet sind, erreicht hat.
- **reachable** Die Dependency ist erfüllt wenn der Vorgänger den als unreachable gekennzeichneten State erreicht hat.

<u>UNRESOLVED</u> Die unresolved Eigenschaft spezifiziert wie das System die Situation behandeln soll, wenn keine Submitted Entity Instanz für ein benötigtes Scheduling Entity, während einer Submit Operation, gefunden wird.

In der folgenden Tabelle werden die möglichen Verhaltensweisen beschrieben:

unresolved	Beschreibung
error	Die submit Operation schlägt fehl mit einer Fehler-
	meldung
ignore	Die Abhängigkeit wird stillschweigend ignoriert
suspend	Die Abhängigkeit wird ignoriert, aber der abhängige
	Submitted Entity wird in 'suspended' gesetzt und
	benötigt eine Benutzer Aktion um fortzufahren

rerun program Wenn eine rerun programm Kommandozeile für einen Job definiert wurde, wird diese Kommandozeile anstelle der run Kommandozeile bei einem Neustart des Jobs, nach einem failure, ausgeführt.

Für Details in Bezug auf commandline parsing, Ausführungen und Parameter substitution siehe die "run program" Option auf Seite 118.

resource Die "resource" section einer Job Definition definiert Resource-Anforderungen zusätzlich zu diesen Anforderungen die indirekt durch die "environment" und "footprint" Optionen definiert wurden.

Wenn hier die selbe Named Resource wie im Footprint angefordert wird, überschreibt die Anforderung in der Resource Section die Anforderung in dem Footprint.

Da environments nur Named Resources mit der usage "static" benötigen und footprints nur Named Resources mit der usage "system", ist die Resource Section einer Job Definition der einzige Platz um Resource-Anforderungen, für Named Resources mit der usage "synchronizing", zu definieren.

Resource-Anforderungen werden durch den vollqualifizierten Pfadnamen zu einer Named Resource mit dem folgenden zusätzlichen Anforderungsoptionen definiert:

AMOUNT Die amount Option ist nur mit Anforderungen für Named Resources von der Art "System" oder "Synchronizing" gültig. Der Amount in einer Resource-Anforderung drückt aus, wieviele Einheiten von der Required Resource belegt werden.

EXPIRED Die expired Option ist nur für Synchronizing Resources mit einem definierten Resource State Profile gültig. Ist die expired Option spezifiziert, darf der Zeitpunkt in der der Resource State von der Resource gesetzt wurde nicht länger her sein als die expire Option angibt. Nun hat eine negative Expiration zur Folge, dass eine Resource mindestens so alt sein muss, wie angegeben. Der Resource State kann nur von dem alter Resource Kommando gesetzt werden (siehe Seite 60) oder automatisch beim definieren eines Resource State Mappings, welcher den Exit State und Resource State in einem neuen Resource State umwandelt. Selbst wenn in so einem Fall der neue Resource State dem alten Resource State gleicht, gilt der Resource State als gesetzt.

<u>LOCKMODE</u> Die lockmode Option in einer Resource-Anforderung ist nur für Synchronizing Resources gültig. Es sind fünf mögliche Lockmodes definiert:

Name	Bedeutung
X	Exclusive lock
S	Shared lock
SX	Shared exclusive lock
SC	Shared compatible lock
N	Nolock

Wichtig ist die Kompatibilitäts Matrix:

	X	S	SX	SC	N
X	N	N	N	N	Y
S	N	Y	N	Y	Y
SX	N	N	Y	Y	Y
SC	N	Y	Y	Y	Y
N	Y	Y	Y	Y	Y

Das Ziel von dem exclusive lock ist, die Resource exclusiv zu haben und um fähig zu sein den Resource State und eventuell Parameter Werte zu setzen. Ein häufiges Beispiel für das Benutzen des exclusive lock ist das neue Laden einer Datenbanktabelle.

Das Ziel vom dem shared lock ist anderen zu erlauben die Resource auf die gleiche Weise zu nutzen, aber um Änderungen zu verhindern. Der gebräuchlichste Beispiel für das Benutzen der shared locks ist einen großer laufender Leser einer Datenbanktabelle. Andere lesende Prozesse können einfach toleriert werden, aber es werden keine schreibenden Transaktionen erlaubt.

Das Ziel von dem shared exclusive lock ist es ein zweites shared lock zu haben, welcher nicht kompatibel mit dem normalen shared lock ist. Wenn wir den normalen shared lock für große lesende Transaktionen benutzen, benutzen wir den shared exclusive lock für kleine schreibende Transaktionen. Kleine schreibende Transaktionen können problemlos parallel zueinander laufen, doch laufen sie parallel zu einer großen lesenden Transaktion führen sie fast sicher zu einem "Snapshot too old" oder zu anderen ähnlichen Problemen.

Das Ziel von dem shared compatible lock ist es ein Typ von shared lock zu haben, welcher sowohl zu dem shared als auch zu dem shared exclusive lock kompatibel ist. Dieser lock-Typ ist für kurze lesende Transaktionen gedacht, welche keine Konflikte mit kleinen schreibenden Transaktionen oder mit großen lesenden Transaktionen haben. Selbstverständlich haben kleine lesende Transaktionen keine Konflikte mit anderen kleinen lesenden Transaktionen. Das parallele Laufen von kleine lesende und große schreibende Transaktionen kann eventuell zu Problemen führen.

Das Ziel von dem nolock ist zu gewährleisten das die Resource existiert und alle anderen Eigenschaften der Resource den Bedarf decken. Die Resource ist nicht locked und alles kann passieren, einschließlich Statusänderungen.

STATE Die state Option ist nur für Synchronizing Resources mit einem Resource State Profile gültig. Die Option wird benutzt um gültige Resource States für diesen Job zu spezifizieren. Eine Resource kann nur allokiert werden, wenn sie in eine von dem angeforderten States ist.

STATE MAPPING Die state mapping Option ist nur für Synchronizing Resources, die ein Resource State Profile spezifizieren und mit einem "exclusive" Lockmode angefordert werden, gültig. Das Mapping definiert eine Funktion die Kombinationen von Exit States und Resource States in einem neuen Resource State abbilden. Für ausführliche Informationen über Resource State Mappings siehe das Create Resource State Mapping Statement auf Seite 129.

KEEP Die keep Option in einer Resource-Anforderung definiert den Zeitpunkt zu dem die Resource freigegeben wird. Die keep Option ist sowohl für System als auch für Synchronizing Resources gültig. Es gibt drei mögliche Werte. Ihre Bedeutung wird in der folgenden Tabelle erklärt:

create job definition

Wert	Bedeutung
nokeep	Die Resource wird am Jobende freigegeben. Dies ist
	das Default-Verhalten
keep	Die Resource wird so bald der Job den Final State erreicht hat freigegeben
keep final	Die Resource wird freigegeben, wenn der Job und alle
	seine Children final sind

STICKY Die sticky Option ist nur für Synchronizing Resources gültig. Wenn sticky spezifiziert ist, wird die Resource in dem Master Batch (dies wird MASTER_RE-SERVATION genannt) allokiert, so lange innerhalb des Batches weitere Jobs existieren die die Resource "sticky" benötigen. Der Amount und Lockmode für die Master Reservation wird aus allen Sticky-Anforderungen aller Children abgeleitet. Der Amount ist das Maximum der Anforderungsmenge.

Der Lockmode ist abhängig von den weiteren Anforderungen. Im Normalfall ist der Lockmode Exclusive, wenn mindestens zwei Jobs vorhanden sind, welche die Resource mit einem anderen Lockmode ungleich Nolock anfordern. Die Ausnahme ist die Kombination von Shared und Shared Compatible Anforderungen. Diese resultiert in einen Lockmode Shared.

Es wird versucht alle Anforderungen aus der Master Reservation zu erfüllen.

Optional kann ein Name für die Sticky Allocation vergeben werden. Es werden grundsätzlich jeweils nur die Anforderungen mit dem selben Namen für das vorher beschriebene Verfahren berücksichtigt. Es kann daher mehrere MASTER_RESER-VATIONS für einen Master Batch gleichzeitig geben. Mit Hilfe der Namen, können innerhalb eines Ablaufs mehrere, von einander getrennte critical Regions realisiert werden.

Zusätzlich oder aber auch alternativ zum Namen kann ein Parent Job oder Batch spezifiziert werden. At Runtime wird über die Submission Hierarchie dann die zugehörige Instanz des Parents ermittelt. Die Sticky Anforderung gilt nur ab dem Parent abwärts. Prinzipiell kann dies so interpretiert werden, als würde die Id des Parents einen Teil des Namens der Sticky Anforderung darstellen. Mit diesem Mechanimus können auf einfache Weise getrennte critical Regions in dynamisch submittete Teilabläufe realisiert werden.

runtime Die runtime Option wird benutzt um die geschätzte Laufzeit eines Jobs zu definieren. Diese Zeit kann beim auslösen von Trigger ausgewertet werden.

run program Die run programm Kommandozeile ist obligatorisch für Jobs, da er das auszuführende Kommando für diese Jobs spezifiziert.

Die Kommandozeile ist in einem Kommando und einer Liste mit Argumenten durch whitespace Charakters getrennt. Das erste Element der Kommandozeile wird als Name des auszuführenden ablauffähigen Programms betrachtet und der Rest als Parameter des Programms.

Ob der Jobserver die Umgebungsvariable PATH beim Suchen nach der ausführbaren Datei benutzt, ist eine Eigenschaft des Jobservers.

System- und Jobparameter können mit \$ Notation angesprochen werden.

Mittels Quoting können whitespace Characters sowie \$-Zeichen als Teil der Kommandozeile weitergeleitet werden. Das Quoting befolgt die Unix Bourne Shell Regeln. Das bedeutet, doppelte Quotes verhindern, dass whitespace Charakters als Trennzeichen interpretiert werden. Einzelne Quotes verhindern auch die Auflösung von Variablen. Es ist möglich Backticks für das Quoting zu verwenden. Teile der Kommandozeile die von Backticks gequoted wurden, werden als einzeln gequoted betrachtet, aber die Backticks bleiben ein Teil des Arguments. Andere Quotes werden entfernt. Sollen Backticks ohne ihre spezielle Bedeutung in der Kommandozeile vorkommen, müssen diese entwertet (mit '\') werden. Beispiel:

Die run Kommandozeile 'sh -c "example.sh \${JOBID} \\$HOME" '\$SHELL'' wird das Programm 'sh' mit den Parameter '-c', 'example.sh 4711 \$HOME' und '\$SHELL' ausführen (in der Annahme, dass das Submitted Entity die Id 4711 hat).

Ist das auszuführenden Programm (erstes Element der Kommandozeile) eine gültige Ganzzahl, so wird die Kommonadozeile nicht vom Jobserver ausgeführt, sondern der Job so behandelt, als hätte er sich mit der Ganzzahl als Exit Code beendet. Dummy Jobs mit 'true' oder 'false' als Programm können nun als '0' statt 'true' bzw. '1' statt 'false' implementiert und so vom System wesentlich effizienter und schneller verarbeitet werden.

Sollte es tatsächlich einmal nötig sein ein Executable mit einer Zahl als Namen auszuführen, so kann dies durch einen Pfad Prefix ('./42' statt '42') erreicht werden.

suspend Die suspend Option definiert ob ein Submitted Entity zur Submit-Zeit suspended wird.

Wenn die suspend Option spezifiziert wird, kann optional die resume Klausel verwendet werden. Damit kann ein automatischer Resume zum angegebenen Zeitpunkt, bzw. nach der angegebenen Zeit bewirkt werden.

Wenn der Resume Zeitpunkt mittels des unvollständigen Datumsformates (siehe dazu auch Seite 6) spezifiziert wird, erfolgt der Resume zum ersten passenden Zeitpunkt nach dem Submitzeitpunkt.

Wenn etwa ein Submit um 16:00 erfolgt, und als Resumezeit ist T17:30 eingetragen, wird der Resume am gleichen Tag um 17:30 erfolgen. Wurde jedoch als Resumezeit T15:55 angegeben, wird der Job bis zum nächsten Tag 15:55 warten müssen.

timeout Die timeout Klausel einer Job Definition definiert die maximale Zeit die ein Job wartet, bis seine Resource Abhängigkeiten erfüllt sind.

Wenn die Timeout-Bedingung erreicht ist, bekommt der Job die exit state die in der Timeout Klausel spezifiziert wurde. Diese Exit State muss Bestandteil des Exit State

create job definition

Profiles sein.

Wenn keine Timeout-Option gegeben ist, wird der Job warten bis alle Anforderungen erfüllt sind.

type Die type Option spezifiziert den Typ des Scheduling Entity der erstellt oder geändert wird.

workdir Die workdir eines Jobs des Typs Scheduling Entity definiert das Verzeichnis in dem das run, rerun oder kill programm ausgeführt werden.

master Die master Option definiert ob dieses Scheduling Entity submitted werden kann um einen Master Run zu erstellen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create named resource

Zweck

Das create named resource Statement wird eingesetzt um eine Klasse von Re- Zweck sourcen zu definieren.

Syntax

Die Syntax des create named resource Statements ist

Syntax

```
create [ or alter ] named resource resourcepath
with WITHITEM {, WITHITEM}
WITHITEM:
     group = groupname [ cascade ]
    inherit grant = none
    inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     parameter = none
     parameter = ( PARAMETER {, PARAMETER} )
    state profile = < none | rspname >
    usage = RESOURCE_USAGE
PRIVILEGE:
     create content
     drop
     edit
    execute
     monitor
     operate
     resource
     submit
     use
     view
PARAMETER:
     parametername constant = string
    parametername local constant [ = string ]
  parametername parameter [ = string ]
RESOURCE_USAGE:
```

create named resource

category | static | synchronizing | system

Beschreibung

Beschreibung

Das create named resource Statement wird benutzt um Klassen von Resources zu definieren. Diese Klassen definieren den Namen, den Usage Type und wahlweise das benutzte Resource State Profile sowie die Parameter.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

parameter Es kann nützlich sein Parameter in Zusammenhang mit der Belegung von Resources zu benutzen. Zum Beispiel könnte eine Resource wie RESOURCE.TEMP_SPACE einen Parameter namens LOCATION haben. Auf diese Weise kann ein Job so eine Resource benutzen und temporären Speicherplatz an einer Stelle, die von der aktuellen Instanz der Named Resource abhängt, allokieren.

Es existieren drei Typen von Parameter in einem Resource Kontext:

Тур	Bedeutung	
constant	Dieser Typ Parameter definiert den Wert, welcher für alle Resources konstant ist.	
local constant	Dieser Typ Parameter definiert einen nicht variablen Parameter, dessen Wert zwischen Instanzen der selben	
parameter	Named Resource abweichen kann. Der Wert eines solchen Parameters kann durch Jobs, die diese Resource exclusiv gesperrt haben, geändert werden.	

Tabelle 6.2.: Named Resource Parameter Typen

state profile Im Fall von Synchronizing Resources kann ein Resource State Profile spezifiziert werden. Dieses erlaubt Jobs die Resource in einem bestimmten State anzufordern. Resource State Änderungen können genutzt werden um Trigger auszulösen.

usage Die Usage der Named Resource kann eine der folgenden sein:

Usage	Bedeutung
category	Kategorien verhalten sich wie Folder und können benutzt werden um die Named Resources in eine übersichtliche Hi- erarchie einzuordnen.
static	Static Resources sind Resourcen die, falls angefordert, in dem Scope in dem der Job läuft vorhanden sein müssen, aber nicht verbraucht werden können. Mögliche Beispiele von Static Resources sind ein bestimmtes Betriebssystem, shared libraries für DBMS-Zugriffe oder das Vorhanden- sein eines C-Compilers.
system	System Resources sind Resources die gezählt werden können. Mögliche Beispiele sind die Anzahl Prozesse, die Menge der Zwischenspeicher oder die Verfügbarkeit von (eine Anzahl von) Bandlaufwerken.
synchronizing	Synchronizing Resources sind die komplexesten Resources und werden benutzt um den Mehrfachzugriff zu synchronizieren. Ein mögliches Beispiel ist eine Datenbank Tabelle. Abhängig von der Art des Zugriffs (große lesende Transaktionen, große schreibende Transaktionen, mehrere kleine schreibende Transaktionen,) kann der Mehrfachzugriff toleriert werden oder auch nicht.
pool	Named Resources vom Typ pool werden benutzt um soge- nannte Resource Pools anzulegen. Diese Pools bieten die Möglichkeit die Verteilung von Amounts für System Re- sources zentral und flexibel zu regeln.

Tabelle 6.3.: Named Resource Usage

factor Beim Anlegen einer Named Resource kann der Faktor, mit der die Mengenangaben in einer Ressourcenanfrage multipliziert werden, spezifiziert werden. Dieser Faktor ist per Default 1. Bei jeder Instanz dieser Named Resource, jede Resource also, kann der Faktor überschrieben werden.

inherit grant Die **inherit grants** Klausel ermöglicht es zu definieren welche Privilegien über die Hierarchie geerbt werden sollen. Wird dieser Klausel nicht spezifiziert, werden per Default alle Rechte geerbt.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

create resource

create resource

Zweck

Zweck

Das create resource Statement wird eingesetzt um eine Instanz von Named Resourcen innerhalb eines Scopes, Folders oder einer Job Definition zu erstellen.

Syntax

Syntax

Die Syntax des create resource Statements ist

```
create [ or alter ] resource resourcepath in < serverpath | folderpath >
with withitem {, withitem}

WITHITEM:
    amount = < infinite | integer >
    | < online | offline >
    | parameter = none
    | parameter = ( PARAMETER {, PARAMETER} )
    | requestable amount = < infinite | integer >
    | state = statename
    | touch [ = datetime ]
    | group = groupname

PARAMETER:

parametername = < string | default >
```

Beschreibung

Beschreibung

Das create resource Statement wird benutzt um Named Resources innerhalb von Scopes, Folders oder Job Definitions zu instanziieren. Im letzteren Fall wird nur ein Template angelegt, welcher materialisiert wird, so bald der Job submitted wird und automatisch zerstört wird so bald der Master Run Final oder Cancelled ist. Ist die **or alter** Option spezifiziert, wird eine bereits existierende Resource geändert, andernfalls wird es als Fehler betrachtet wenn die Resource bereits existiert.

amount Die **amount** Klausel definiert den Available Amount von dieser Resource. Im Falle von statischen Resources, wird die Amount-Option nicht spezifiziert.

base multiplier Der base Multiplier ist nur relevant wenn das Resource Tracing genutzt wird. Der base Multiplier bestimmt den Multiplikationsfaktor von **trace base**. Wenn der trace Base mit B und der trace Multiplier mit M bezeichnet wird gilt, dass über die Zeiträume $B*M^0$, $B*M^1$ und $B*M^2$ die Durschschnittsbelegung

ermittelt wird. Der Default ist 600 (10 Minuten), so dass die Werte für B, 10B und 100B (in Minuten) ermittelt werden

factor Um Resource Anforderungen von Jobs von aussen justieren zu können, wurde ein Resource Factor eingeführt. Dieser kann sowohl an der Named Resource als auch individuell an der Resource gesetzt werden. Bei der Bestimmung ob ein Job eine bestimmte Resource belegen kann wird durch den Vergleich der ursprünglichen Anforderung mit dem Requestable Amount bestimmt. Bei der tatsächlichen Belegung wird jedoch

ceil(Anforderung * Factor)

hergenommen.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

online Die **online** Klausel definiert, ob die Resource online oder offline ist.

Wenn eine Resource offline ist, steht sie nicht zur Verfügung. Das bedeutet, dass ein Job der diese Resource benötigt nicht innerhalb dieses Scopes laufen kann. Doch da die Resource Online gesetzt werden kann, wird der Job warten und nicht in einen Fehlerstatus versetzt werden.

Das ist ebenfalls für statische Resources gültig.

parameter Die **parameter** Klausel wird benutzt um die Werte von den Parameter, wie sie für die Named Resource definiert wurden, zu spezifizieren.

Parameter die als Konstante auf Named Resource Level deklariert sind, sind hier nicht erlaubt. Alle anderen Parameter können, müssen aber nicht, spezifiziert werden. Wenn kein Parameter und kein Default, für den Parameter auf dem Named Resource Level, spezifiziert ist, wird bei der Auflösung ein leerer String zurückgegeben.

Wird beim Ändern der Resource der Parametername = default spezifiziert, hat dies zur Folge, dass der Wert des Parameters den default Wert, so wie bei der Named Resource spezifiziert, erhält.

Wenn der Parameter auf der Named Resource Ebene geändert wird, wird dies auf der Resource Ebene für alle Parameter die auf Default gesetzt wurde sichtbar sein. Es sind für jede Resource immer eine Anzahl von System Variablen definiert. Diese werden vom System gesetzt und stehen Jobs, welche die Ressource allokieren über "RESSOURCEREFERENCES" für einen lesenden Zugriff zur Verfügung.

Diese System Variable sind:

User Commands create resource

Name	Description
STATE	Der Resource Status einer "synchronizing" Ressource mit einem Status Modell.
AMOUNT	Der insgesamt zur Verfügung stehende Ressourcen Betrag
FREE_AMOUNT	Der freie zur Verfügung stehende Ressourcen Betrag
REQUESTABLE_AMOUNT	Der von einem Job maximal allokierbare Betrag
REQUESTED_AMOUNT	Der vom Job angeforderte Betrag
TIMESTAMP	Touch Zeitstempel einer "synchronizing" Ressource mit einem Status Modell.

Tabelle 6.4.: List of System Variables

requestable amount Die **requestable amount** Klausel definiert den Amount von dieser Resource, die von einem einzelnen Job angefordert werden darf. Dieses kann von dem available Amount verschieden sein. Wenn die angeforderte Menge kleiner als der Amount ist, ist es sicher, dass ein Job nicht alle verfügbaren Resources allokieren kann. Wenn der requestable Amount grösser als der Amount ist, können Jobs mehr anfordern als an Amount zur Verfügung steht, ohne ein "cannot run in any Scope" Fehler zu erzeugen.

Wenn der Requestable Amount nicht spezifiziert ist, gleicht er dem Amount. Im Falle von statischen Resources, wird die requestable Amount Option nicht spezifiziert.

state Die **state** Klausel definiert den Status in dem die Resource ist. Diese Option ist nur für synchronizing Resources mit einem Resource State Profile gültig.

tag Um die Trace Tabelle leichter auswerten zu können, können Resources und Pools nun mit einem Tag versehen werden. Dieser Tag soll innerhalb Resources und Pools eindeitig sein. (D.h. auch das Benutzen eines Tags für sowohl eine Resource als auch einen Pool ist verboten).

touch Die **touch** Klausel definiert den letzten Zeitpunkt in dem der Status der Resource (von einem Job) geändert wurde. Dieser Timestamp wird nicht gesetzt wenn ein Resource State manuell gesetzt wurde.

Diese Option ist nur für synchronizing Resources, mit einem Resource State Profile, gültig.

trace base Falls der trace base **none** ist, ist Tracing ausgeschaltet. Ansonsten ist es die Basis für den Auswertungszeitraum.

create resource

User Commands

trace interval Das trace Interval ist die minimale Zeit zwischen dem Schreiben von Trace Records in Sekunden. Ist das trace Interval **none**, ist Tracing ausgeschaltet.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

create resource state definition

create resource state definition

Zweck

Zweck

Das create resource state definition Statement wird eingesetzt um einen symbolischen Namen für den Resource State zu erstellen.

Syntax

Syntax

Die Syntax des create resource state definition Statements ist

create [**or alter**] **resource state definition** *statename*

Beschreibung

Beschreibung

Das create resource state definition Statement wird benutzt um einen symbolischen Namen für einen Resource State zu definieren.

Das optionale Schlüsselwort **or alter** wird benutzt um das Auftreten von Fehlermeldungen und das Abbrechen der laufenden Transaktion, wenn eine Resource State Definition bereits existiert, zu verhindern. Ist es nicht spezifiziert, führt die Existenz einer Resource State Definition mit dem spezifizierten Namen zu einem Fehler.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Beispiel

Beispiel

In diesen Beispielen werden eine Anzahl Namen für Resource States definiert.

```
create resource state definition leer;

create resource state definition gueltig;

create resource state definition ungueltig;

create resource state definition stadium1;

create resource state definition stadium2;

create resource state definition stadium3;
```

create resource state mapping

Zweck

Das create resource state mapping Statement wird eingesetzt um ein Mapping Zweck zwischen den Exit States eines Jobs und dem resultierenden Resource State einer Resource zu definieren.

Syntax

Die Syntax des create resource state mapping Statements ist

Syntax

```
create [ or alter ] resource state mapping mappingname
with map = ( WITHITEM \{, WITHITEM\} )
```

WITHITEM:

statename maps < statename | any > to statename

Beschreibung

Das create resource state mapping statement definiert das Mapping von Exit States in Kombination mit Resource States zu neuen Resource States.

Beschreibung

Der erste State-Name muss ein Exit State sein. Der zweite und dritte State jeweils ein Resource State. Terminiert ein Job mit dem genannte Exit State, wird der Status des Resources auf den neuen Status gesetzt, wenn der aktuelle Status mit dem erstgenannten Status übereinstimmt. Wird als Anfangsstatus any spezifiziert, wird jede beliebige Resource States auf den neuen abgebildet. Wenn sowohl ein spezifisches Mapping als auch ein generelles Mapping spezifiziert sind, hat das spezifische Mapping die höchste Priorität.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Beispiel

Beispiel

create resource state profile

create resource state profile

Zweck

Zweck

Das create resource state profile Statement wird eingesetzt um eine Anzahl von gültigen Resource States zu erstellen.

Syntax

Syntax

Die Syntax des create resource state profile Statements ist

```
create [ or alter ] resource state profile profilename
with WITHITEM {, WITHITEM}

WITHITEM:
    initial state = statename
    | state = ( statename {, statename} )
```

Beschreibung

Beschreibung

Das create resource state profile Statement wird benutzt um eine Menge von gültige Resource States für eine (Named) Resource zu definieren.

state Die **state** Klausel definiert welche Resource State Definitions innerhalb dieses Profils gültig sind.

initial state Die initial state Klausel bestimmt den initialen State einer Resource mit diesem Profile. Der initial State muss nicht in der Liste der States aus der **state** Klausel enthalten sein. Dies erlaubt das Anlegen einer Resource, ohne dass diese Resource sofort eine Aktive Rolle im System spielt.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Beispiel

Beispiel

In diesem Beispiel soll der Exit Status leer ungültig werden.

```
create resource state profile example1
with
   state = (leer);
```

create schedule

User Commands

create schedule

Zweck

Das create schedule Statement wird eingesetzt um einen aktiven Container für Zweck Scheduled Events zu erstellen.

Syntax

Die Syntax des create schedule Statements ist

Syntax

```
create or alter schedule schedulepath with withitem withitem
WITHITEM:
     < <u>active</u> | inactive >
  inherit grant = none
    inherit grant = ( PRIVILEGE {, PRIVILEGE} )
    interval = < none | intervalname >
     time zone = string
     group = groupname
```

PRIVILEGE: create content drop edit execute monitor operate resource submit use view

Beschreibung

Mit dem create schedule Statement kann man mittels einfacher Definitionen Beschreibung komplexe Zeitpläne für Jobs und Batches erstellen.

active Die Angabe active bewirkt, dass der Schedule im Takt des spezifizierten Intervalls prinzipiell Ereignisse auslöst (vorausgesetzt, es sind welche definiert). Die Angabe inactive bewirkt dagegen, dass der Schedule im Takt des spezifizierten Intervalls gerade das Auslösen von Ereignisse verhindert. Durch die hierarchische Anordnung von Schedules können auf diese Weise etwa Ausnahmeperioden (wie Downtimes) gebildet werden.

create schedule

Group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

Interval Das angegebene Intervall fungiert als Taktgeber für den Schedule. Wird ein Event mit dem Schedule verknüpft, wird dieser Event im Rhythmus des Intervalls ausgelöst.

inherit grant Die **inherit grants** Klausel ermöglicht es zu definieren welche Privilegien über die Hierarchie geerbt werden sollen. Wird dieser Klausel nicht spezifiziert, werden per Default alle Rechte geerbt.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create scheduled event

Zweck

Der Zweck von dem create scheduled event Statements ist, eine Verbindung Zweck zwischen einem Event und einem Zeitplan zu definieren.

Syntax

Die Syntax des create scheduled event Statements ist

Syntax

Beschreibung

Scheduled Events stellen eine Verbindung zwischen Events (was ist zu tun) und Beschreibung Schedules (wann soll es gemacht werden) dar.

cwbacklog handling Das Backlog Handling gibt an, wie mit Events, die in Zeiten in den der Server down war aufgetreten sind, umgegangen werden soll. In der untenstehende Tabelle sind die drei Möglichkeiten aufgeführt:

Möglichkeit	Bedeutung
last	Nur der letzte Event wird ausgelöst
all	Alle zwischenzeitlich eingetretene Events werden ausgelöst
none	Keiner der zwischenzeitlich eingetretene Events wird ausgelöst

Group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

active Scheduled Events können als **active** oder **inactive** gekennzeichnet werden. Wenn sie als active gekennzeichnet sind, werden Events ausgelöst. Dementsprechend

create scheduled event

werden Events nicht ausgelöst, wenn der Scheduled Event als inactive gekennzeichnet ist. Mittels dieser Option können Scheduled Events deaktiviert werden, ohne dass die Definition verloren geht.

suspend limit Das Suspend Limit gibt an, nach wieviel Verspätung ein zu einem Event gehörende Job automatisch mit der Suspend-Option submitted wird. Eine Verspätung kann auftreten wenn, aus welchem Grund auch immer, der Scheduling Server für einige Zeit down ist. Nach dem Hochfahren des Servers werden die zwischenzeitliche Events, abhängig von der **backlog handling** Option, ausgelöst. Die Ausführungszeit ist damit später als die geplante Ausführungszeit.

Ausgabe

Ausgabe

create scope

User Commands

create scope

Zweck

Das create scope Statement wird eingesetzt um einen Scope innerhalb der Scope- Zweck Hierarchie zu erstellen.

Syntax

Die Syntax des create scope Statements ist

Syntax

```
create [ or alter ] < scope serverpath | job server serverpath > [ with
JS_WITHITEM {, JS_WITHITEM} ]
JS_WITHITEM:
     config = none
    config = ( CONFIGITEM {, CONFIGITEM} )
     < enable | disable >
     error text = < \underline{none} \mid string >
     group = groupname [ cascade ]
     inherit grant = none
     inherit grant = ( PRIVILEGE {, PRIVILEGE} )
     node = nodename
     parameter = none
     parameter = ( PARAMETERITEM {, PARAMETERITEM} )
     password = string
     rawpassword = string [ salt = string ]
CONFIGITEM:
     parametername = none
    parametername = ( PARAMETERSPEC {, PARAMETERSPEC} )
  | parametername = < string | number >
PRIVILEGE:
     create content
   drop
     edit
     execute
     monitor
     operate
    resource
```

create scope

submit use view

PARAMETERITEM:

```
parametername = dynamic
| parametername = < string | number >
```

PARAMETERSPEC:

parametername = < string | number >

Beschreibung

Beschreibung

Das create scope Kommando wird benutzt um einen Scope oder Jobserver und seine Eigenschaften zu definieren.

Config Die config Option erlaubt die Konfiguration eines Jobservers mittels Key/Value Pairs. Die Konfiguration wird nach unten vererbt, so dass generelle Konfigurations Parameter bereits auf Scope-Ebene gesetzt werden können und damit für alle darunter angelegte Jobserver ihre Gültigkeit haben, sofern die Parameter auf unterer Ebene nicht überschrieben werden.

Bei der Anmeldung eines Jobservers wird diesem die Liste mit Konfigurations Parameter übergeben.

Enable Die enable Option erlaubt dem Jobserver die Verbindung zu dem Repository Server. Diese Option ist für Scopes ungültig und wird, wenn sie spezifiziert wird, stillschweigend ignoriert.

Disable Die disable Option verbietet dem Jobserver die Verbindung zum Repository Server. Diese Option ist für Scopes ungültig und wird, wenn sie spezifiziert wird, stillschweigend ignoriert.

Group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

Node Der Node gibt an, auf welchem Rechner der Jobserver läuft. Dieses Feld hat einen rein dokumentativen Charakter.

Parameter Parameter können zur Kommunikation und Datenübermittlung zwischen Jobs verwendet werden. Sie stehen den Jobs und den Programmen, welche innerhalb der Jobs ausgeführt werden, zur Verfügung.

Die Parameter von Scopes und Jobservers können dazu benutzt werden Information über die Laufzeitumgebung eines Jobs zu spezifizieren.

Bei dem Dynamic Parameter wird der Parameter nach der Anmeldung des Jobservers aus seiner eigenen Prozessumgebung gefüllt. Beim Ändern der Prozessumgebung eines Jobservers muss auf diese Dynamic Variable geachtet werden, da ansonsten leicht race Conditions entstehen.

Password Die password Option wird benutzt, um das Passwort des Jobservers zu setzen. Diese Option ist für Scopes ungültig und wird, wenn sie spezifiziert wird, stillschweigend ignoriert.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

create trigger

create trigger

Zweck

Zweck

Das create trigger Statement wird eingesetzt um ein Objekt welches einen Job dynamisch submitted, wenn eine bestimmte Kondition gegeben ist, zu erstellen.

Syntax

Syntax

Die Syntax des create trigger Statements ist

```
create [ or alter ] trigger triggername on CT_OBJECT [ < noinverse |
inverse > ]
with WITHITEM {, WITHITEM}
CT_OBJECT:
     job definition folderpath
WITHITEM:
     < <u>active</u> | inactive >
     < nowarn | warn >
     limit state = < none | statename >
     < <u>nomaster</u> | master >
     < <u>noresume</u> | resume in period | resume at datetime >
     state = none
     state = ( < statename {, statename} |</pre>
     CT_RSCSTATUSITEM {, CT_RSCSTATUSITEM} > )
     submit after folderpath
     submit folderpath
     submitcount = integer
     < nosuspend | suspend >
     [type = ] CT_TRIGGERTYPE
     group = groupname
CT EVENT:
< create | change | delete >
CT_RSCSTATUSITEM:
< statename any | statename statename | any statename >
```

CT_TRIGGERTYPE:

before final
| immediate local
| immediate merge

warning

.

Beschreibung

Das create trigger Statement wird benutzt um ein Objekt zu erstellen, welches darauf wartet, dass ein bestimmtes Ereignis eintritt, nach welches, als Reaktion auf diesem Event, ein Job oder Batch submitted wird.

Beschreibung

Ist die **or alter** Option spezifiziert, wird ein bereits existierender Trigger geändert, andernsfalls führt es zu Fehler, wenn der Trigger bereits existiert.

Trigger können für Scheduling Entities oder Synchronizing (Named) Resources definiert werden. Im letzteren Fall wird der Trigger bei jedem Statuswechsel der Resource oder Instanz der Named Resource ausgewertet. Resource Trigger werden immer sogenannte Master Trigger sein, das bedeutet, sie submitten einen neuen Master Batch oder Job. Trigger in Scheduling Entities können Master Batches submitten, aber submitten standardmäßig neue Kinder. Diese Kinder müssen als (dynamische) Kinder des triggernden Scheduling Entities definiert sein.

active Die active Option ermöglicht das aktivieren beziehungsweise deaktivieren des Triggers. Damit kann das Triggern vorübergehend unterbunden werden, ohne den Trigger löschen zu müssen.

check Die check Option ist nur für **until final** und **until finished** Trigger gültig. Es definiert die Zeitintervalle zwischen zwei Auswertungen der Bedingungen. Ohne Rücksicht auf die definierten Intervalle zu nehmen, wird die Condition auf jeden Fall überprüft, wenn ein Job beendet wird.

condition Die condition Option kann spezifiziert werden um eine zusätzliche Bedingung, welche geprüft werden muss bevor der Trigger feuert, zu definieren. Diese Bedingung ist ein boolscher Ausdruck und das Feuern findet statt wenn diese Bedingung true ergibt.

<u>BOOLSCHE OPERATOREN</u> Da diese Bedingung ein boolscher Ausdruck ist, können boolsche Operatoren benutzt werden um mehrere komplexe Bedingungen zu erstellen. Diese boolsche Operatoren sind:

- **not** (unärer Negations Operator)
- and
- or

Dabei gelten die üblichen Priotitätsregeln. Der not-Operator hat vor dem and-Operator Vorrang, welcher vor dem or-Operator Vorrang hat. Es ist erlaubt Klammern zu benutzen um eine Auswertungsreihenfolge zu erzwingen.

Ferner ist es erlaubt die boolesche Konstante false und true zu benutzen.

<u>VERGLEICHS OPERATOREN</u> Vergleiche können als Teil von boolesche Ausdrücke benutzt werden. Folgende Vergleichsoperatoren werden definiert.

- == (gleich)
- >= (größer oder gleich)
- <= (kleiner oder gleich)
- ! = (ungleich)
- > (größer als)
- < (kleiner als)
- = \sim (pattern matches)
- $! \sim (pattern matches nicht)$

Alle Vergleichsoperatoren können mit Zeichenketten arbeiten. Die größer und kleiner als Operatoren benutzen im Fall von Zeichenketten den ascii-Wert von den Charaktern. Die matching-Operatoren arbeiten nicht mit Zahlen.

Für eine vollständige Beschreibung der regulären Ausdrücke, welche von den match-Operatoren benutzt werden können, verweisen wir auf die originale Java Dokumentation der java.util.regexp.

<u>NUMERISCHE OPERATOREN</u> Da nicht garantiert werden kann das Entscheidungen nicht nur von den Abgleich zweier Werte getroffen werden kann, ist es erlaubt (numerische) Operatoren zu benutzen. Die gültigen Operatoren sind:

- + (unärer Operator)
- – (unärer Negation Operator)
- * (multiplications Operator)
- / (divisions Operator)
- % (modulo Operator)
- + (binärer Additions Operator)
- – (binärer Subtractions Operator)

<u>LITERALE UND VARIABLEN</u> Literals sind Zahlen (ganze Zahlen und Fließkomma Zahlen) oder Zeichenketten. Zeichenketten werden durch doppelte quotes (") abgegrenzt. Es ist möglich Variable zu benutzen, welche innerhalb des Kontext des triggernden Jobs oder Resource aufgelöst werden. Variable werden adressiert, indem man ihren Namen einen Dollarzeichen (\$) voranstellt.

Bei der Auflösung der Variablen wird zuerst angenommen, dass es sich um eine Triggervariable handelt. Trifft dies nicht zu, wird die Variable als Jobvariable interpretiert. Diese Art der Auflösung ist zwar häufig richtig, aber leider nicht immer. Durch das Voranstellen von job., trigger. oder resource. kann explizit angegeben werden bei welchem Objekt nach der Variable gesucht werden soll.

Normalerweise werden Variablen in Großbuchstaben angelegt. Dies kann durch Quoting der Namen verhindert werden. Allerdings wird beim Ansprechen der Variablen in Conditions der Name wieder in Großbuchstaben konvertiert. Um dies zu verhindern, muss der Name sowie einen eventuellen Prefix in geschweiften Klammern geschrieben werden.

Abhängig von dem Operator und dem ersten Operand, werden die Operands als Zeichenketten oder Zahlen interpretiert. Multiplikationen, Divisionen, Modulo und Subtractionen sowie die unären Abläufe sind nur für numerische Werte definiert. Der Additions-Operator in einem Zeichenketten Kontext bewirkt das Aneinanderreihen der Operanden.

<u>FUNKTIONEN</u> Weil nicht alles einfach mittels (numerische) Ausdrücke ausgedrückt werden kann, sind einige Funktionen dazugekommen. Zur Zeit sind folgende Funktionen definiert:

- abs(expression) der absolute Wert des Ausdrucks wird zurückgegeben
- int(expression) der ganzzahlige Wert des Ausdrucks wird zurückgegeben
- **lowercase**(*expression*) das Ergebnis des Ausdrucks wird in Kleinbuchstaben umgewandelt und zurückgegeben
- round(expression) der Ausdruck wird gerundet und zurückgegeben
- str(expression) der Ausdruck wird als Zeichenkette zurückgegeben
- **substr**(*source*, *from* [, *until*]) gibt einen Teil der Zeichenkette *source*, beginnend in der Position *from* bis zum Ende der Zeichenkette, oder wenn *until* spezifiziert ist, bis zur Position *until*, zurück.
- **trim**(*expression*) der Ausdruck wird ohne Leerzeichen am Schluß zurückgegeben.
- **uppercase**(*expression*) das Ergebnis des Ausdrucks wird in Großbuchstaben umgewandelt und zurückgegeben

create trigger

Funktionen können ohne Beschränkung inneinander Verschachtelt werden.

BEISPIELE Zur Verdeutlichung folgen jetzt einige Statements die Conditions spezifizieren. Da Conditions nicht ausschliesslich in der Definition von Triggers vorkommen, gibt es auch andere Beispiele. Die Syntax ist jedoch immer dieselbe. Das erste Beispiel zeigt einen Trigger der dann ausgelöst wird, wenn der Job auf WARNING oder FAILURE geht, aber schon Zeilen verarbeitet hat (\$NUM_ROWS > 0\$).

```
CREATE OR ALTER TRIGGER ON_FAILURE

ON JOB DEFINITION SYSTEM.EXAMPLES.E0100_TRIGGER.TRIGGER

WITH

STATES = (FAILURE, WARNING),
SUBMIT SYSTEM.EXAMPLES.E0100_TRIGGER.ON_FAILURE,
IMMEDIATE MERGE,
ACTIVE,
NOMASTER,
SUBMITCOUNT = 3,
NOWARN,
NOSUSPEND,
CONDITION = '$NUM_ROWS > 0';
```

Das zweite Beispiel zeigt ein Environment, welches fordert, dass der Wert der Resource Variable AVAILABLE mit einem T anfangen soll (wie etwa TRUE, True, true oder Tricky).

Das dritte Beispiel zeigt dasselbe wie das zweite, nur ist der Parametername in mixed Case definiert.

event Die event Option ist nur für Object Monitor Triggers relevant. Sie spezifiziert bei welcher Art von Ereignisse der Trigger gefeuert werden soll.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

main Die master Option wird benutzt um festzulegen ob der Job als Master submitted wird oder nicht. Diese Option hat nur für Job Trigger eine Bedeutung, da Resource Trigger immer als Master submitted werden.

master Die master Option wird benutzt um festzulegen ob der Job als Master submitted wird oder nicht. Diese Option hat nur für Job Trigger eine Bedeutung, da Resource Trigger immer als Master submitted werden.

parent Die parent Option ist nur für Object Monitor Triggers relevant. Sie kann auch nur in Kombination mit der main Option spezifiziert werden.

Wenn sie spezifiziert ist, hat es zur Folge, dass der entsprechende Job (oder Batch) innerhalb des über den main Job submitteden Baumes gesucht wird und die Triggerjobs unter den Parent gehängt werden.

rerun Die rerun Option kann nur reagieren auf restartable States und hat einen automatischen Rerun zur Folge. In vielen Fällen wird es sinnvoll sein auch die suspend/resume Optionen zu spezifizieren um eine gewisse Zeit zwischen den Wiederholungen zu lassen.

Entweder die submit Option, oder die Rerun Option muss spezifiziert werden.

resume Die resume Option kann zusammen mit der suspend Option verwendet werden um eine verzögerte Ausführung zu bewirken. Es gibt dabei zwei Möglichkeiten. Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll.

Bei der unvollständige Angabe eines Zeitpunktes, wie etwa T16:00, wird der Zeitpunkt der Triggerauslösung als Referenzzeit hergenommen.

state Die state Option ist für alle, außer **until final** und **until finished**, Trigger gültig. Im Falle von Trigger auf Jobs, kann eine Liste von Exit States spezifiziert werden. Wenn der Job, in dem der Trigger definiert ist, einen Exit State, welcher in der Trigger Definition gelistet ist, erreicht, dann feuert der Trigger (es sei denn, es ist eine Konditon spezifiziert die mit false bewertet wird).

Im Falle von einem Trigger auf einer (Named) Resource, kann eine Liste von Statuswechsel spezifiziert werden. Auf diesem Weg kann jeder Statuswechsel explizit adressiert werden. Es ist möglich zu triggern, wenn ein Status verlassen wird, durch die Benutztung des Schlüsselwortes any auf der rechten Seite. Es ist jederzeit möglich auf das Erreichen eines bestimmten Status, durch das Spezifizieren von any auf der linken Seite, zu triggern. Um auf jeden Statuswechsel zu triggern, wird die State Option ausgelassen.

submit Die submit Option definiert welcher Job oder Batch submitted wird, wenn der Trigger feuert.

Entweder die submit Option, oder die Rerun Option muss spezifiziert werden.

submitcount Die submitcount Option ist nur bei Trigger auf Jobs zulässig. Es definiert die Anzahl mal die ein Trigger feuern kann. Wenn diese Option nicht spezifiziert ist, wird ein submitcount von 1 genommen.

Wenn ein submitcount von 0 spezifiziert ist, wird der submitcount auf dem Serverparameter TriggerSoftLimit (dessen default-Wert 50 ist) gesetzt. Handelt es sich jedoch um einen rerun-Trigger, bedeutet ein Submitcount von 0, dass es keinen Limit bezüglich der Anzahl Restart Versuchen gibt.

Ist ein submitcount, der grösser als der TriggerSoftLimit ist, spezifiziert, wird der submitcount auf dem Serverparameter TriggerHardLimit (dessen Wert per Default 100 ist) beschränkt. Dies wird gemacht, um endlosschleifen zu vermeiden. Der TriggerHardLimit kann in der Serverkonfiguration auf $2^{31}-1$ gesetzt werden um die obere Schranke praktisch zu eliminieren.

suspend Die suspend Option wird benutzt um den Job oder Batch suspended zu submitten. Diese Option ist für alle Triggertypen gültig.

type Es gibt mehrere Typen von Trigger in Jobs. Die wichtigste Differenz zwischen ihnen ist die Zeit, zu der sie überprüft werden. Die folgende Tabelle zeigt eine Liste von allen Typen mit einer kurzen Beschreibung ihres Verhaltens.

Es muss hervorgehoben werden, dass die Type-Option nicht für (named) resource trigger gültig ist.

Feld	Beschreibung
Тур	Prüfungszeit
after final	Nur nachdem ein final state erreicht wurde, wird überprüft ob der definierte Trigger feuern muss. Wenn der Trigger kein Master Trigger ist, wird der neu submittete Job den gleichen Parent wie der triggernde Job haben. Eine besondere Situation tritt ein, wenn der triggernde Job seinen eigenen submit triggert. In diesem Fall ersetzt der neu submittete Job den triggernden Job. Da dieser Austausch statt findet, bevor die Abhängigkeit überprüft wurde, warten alle abhängigen Jobs, bis der neu submittete Job final ist.
Fortsetzung auf der nächst	ten Seite

Feld	Description
before final	Direkt bevor ein final state erreicht wird, wird überprüft ob der definierte Trigger feuern soll. Das ist die letzte Möglichkeit neue Children zu submitten. Wird das gemacht, dann wird der Job oder Batch zu diesem Zeitpunkt keinen final state erreichen.
finish child	Ein finisch child Trigger prüft jedes mal, wenn ein direktes oder indirektes Child sich beendet, ob gefeuert werden soll.
immediate local	Der immediate local Trigger prüft ob er feuern muss, wenn ein Job terminiert. Nur der exit state des Jobs wird berücksichtigt.
immediate merge	Der immediate merge Trigger prüft ob er feuern muss, so bald der merged exit state wechselt.
until final	Der until final Trigger prüft periodisch ob er feuern muss. Diese Prüfung startet sobald ein Job oder Batch submitted wurde und hält nicht an, bevor er final ist. Der until final Trigger benötigt zwingend eine Condition. Diese Condition wird zumindest einmal geprift. Diese Prüfung findet statt wenn der Job oder Batch in den State finished wechselt.
until finished	Der until finished Trigger ähnelt dem until final trigger. Der einzige Unterschied ist, dass der until finished trigger die Prüfung beendet, sobald der Job finished ist. Der until finished Trigger benötigt zwingend eine Condition. Diese Condition wird zumindest einmal geprift. Diese Prüfung findet statt wenn der Job oder Batch in den State finished wechselt.

create trigger

Fortsetzung von der vorherigen Seite				
Feld	Description			

Tabelle 6.5.: Beschreibung der verschiedenen Trigger Typen

Ausgabe

Ausgabe

create user

User Commands

create user

Zweck

Das create user Statement wird eingesetzt um ein Wertepaar zu erstellen, welches Zweck benutzt werden kann um sich beim Server zu authentifizieren.

Syntax

Die Syntax des create user Statements ist

Syntax

Beschreibung

```
create [ or alter ] user username
with WITHITEM {, WITHITEM}

WITHITEM:
    default group = groupname
    | < enable | disable >
    | group = ( groupname {, groupname} )
    | password = string
    | rawpassword = string [ salt = string ]
```

Beschreibung

Das create user Statement wird benutzt um einen Benutzer anzulegen. Ist "or alter" spezifiziert, wird ein bereits existierender Benutzer geändert. Andernfalls führt ein bereits existierender Benutzer zu einem Fehler.

Die **default group** Klausel wird benutzt um die Default Group zu spezifizieren.

default group Die default group Klausel definiert die Gruppe welche als Eigentümer für alle seine Objekte die der Benutzer erstellt benutzt wird, wenn keine explizite Gruppe bei der Objekterstellung spezifiziert wurde.

enable Die enable Option erlaubt dem Benutzer die Verbindung zu dem Repository Server.

disable Die disable Option verbietet dem Benutzer die Verbindung zum Repository Server.

group Die group Klausel wird benutzt um die Gruppen zu denen der Benutzer gehört zu spezifizieren. Jeder Benutzer ist ein Mitglied der Systemgruppe Public.

create user | 147

create user

password Die password Option wird benutzt, um das Passwort des Users zu setzen.

rawpassword Das rawpassword wird benutzt um das Passwort des Users zu setzten, das nötig ist um mit dem Repository Server verbunden zu werden. Das rawpassword ist das bereits verschlüsselte Passwort. Die rawpassword Option ist für create user Kommandos von dem dump Kommando erzeugt worden.

Ausgabe

Ausgabe

7. deregister commands

deregister

deregister

Zweck

Zweck

Das deregister Statement wird eingesetzt um den Server zu benachrichtigen, das der Jobserver keine Jobs mehr ausführt. Siehe das register Statement auf Seite 248.

Syntax

Syntax

Die Syntax des deregister Statements ist

deregister serverpath . servername

Beschreibung

Beschreibung

Das deregister Statement wird genutzt um den Server über einen, mehr oder weniger, permanenten Ausfall eines Jobservers zu informieren.

Diese Nachricht hat verschiedene Serveraktionen zur Folge. Als erste werden alle running Jobs des Jobservers, d.h. Jobs im Status **started**, **running**, **to_kill** und **killed**, auf den Status **broken_finished** gesetzt. Jobs im Status **starting** werden wieder auf **runnable** gesetzt. Dann wird der Jobserver aus der Liste der Jobservers die Jobs verarbeiten können entfernt, so dass dieser Jobserver im Folgenden auch keine Jobs mehr zugeteilt bekommt. Als Nebeneffekt werden Jobs, die aufgrund Resourceanforderungen nur auf diesen Jobserver laufen können, in den Status **error**, mit der Meldung "Cannot run in any scope because of resource shortage", versetzt. Als letzte wird einen kompletten Reschedule ausgeführt um eine Neuverteilung von Jobs auf Jobservers herbei zu führen.

Durch erneutem Registrieren (siehe das register Statement auf Seite 248), wird der Jobserver erneut in die Liste der Jobs-verarbeitenden Jobservers eingetragen.

Ausgabe

Ausgabe

8. disconnect commands

disconnect

disconnect

Zweck

Zweck Das disconnect Statement wird eingesetzt um die Serververbindung zu beenden.

Syntax

Syntax Die Syntax des disconnect Statements ist

disconnect

Beschreibung

Beschreibung

Mit dem disconnect Statement kann die Verbindung zum Server beendet werden.

Ausgabe

Ausgabe

9. drop commands

drop comment

drop comment

Zweck

Zweck

Das drop comment Statement wird eingesetzt um einen Kommentar zu einem Objekt zu löschen.

Syntax

Syntax

Die Syntax des drop comment Statements ist

drop [existing] comment on OBJECTURL

```
OBJECTURL:
```

distribution distributionname for pool resourcepath in serverpath
environment environmentname
exit state definition statename
exit state mapping mappingname
exit state profile profilename
event eventname

resource resourcepath in folderpath

folder folderpath

footprint footprintname

group groupname

interval intervalname

job definition folderpath

job jobid

named resource resourcepath

parameter parametername of PARAM_LOC

resource state definition statename

resource state mapping mappingname

resource state profile profilename

scheduled event schedulepath . eventname

schedule schedulepath

resource resourcepath in serverpath

< **scope** *serverpath* | **job server** *serverpath* >

trigger triggername on TRIGGEROBJECT [< noinverse | inverse >]

user username

PARAM_LOC:

folder folderpath

job definition folderpath

drop comment

User Commands

| < scope serverpath | job server serverpath >

TRIGGEROBJECT:

resource resourcepath in folderpath
job definition folderpath
object monitor objecttypename
resource resourcepath in serverpath

Beschreibung

Das drop comment Statement löscht den zu dem angegebenen Objekt vorhandenen Kommentar. Wenn das Schlüsselwort **existing** nicht spezifiziert wird, wird das fehlen eines Kommentars als Fehler betrachtet.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop environment

drop environment

Zweck

Zweck

Das drop environment Statement wird eingesetzt um das spezifizierte Environment zu löschen.

Syntax

Syntax

Die Syntax des drop environment Statements ist

drop [existing] environment environmentname

Beschreibung

Beschreibung

Das drop environment Statement wird benutzt um eine Definition von einem Environment zu löschen. Es führt zu einem Fehler, wenn Jobs immer noch dieses Environment benutzen. Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler betrachtet wenn das spezifizierte Environment nicht existiert.

Ausgabe

Ausgabe

drop event

User Commands

drop event

Zweck

Das drop event Statement wird eingesetzt um das spezifizierte Event zu löschen. Zweck

Syntax

Die Syntax des drop event Statements ist

Syntax

drop [existing] event eventname

Beschreibung

Das drop event Statement wird benutzt um eine Definition eines Events zu löschen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Event nicht existiert.

Ein Event kann nicht gelöscht werden wenn es zugehörige Scheduled Events gibt.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Beschreibung

drop exit state definition

drop exit state definition

Zweck

Zweck

Das drop exit state definition Statement wird eingesetzt um die spezifizierte Exit State Definition zu löschen.

Syntax

Syntax

Die Syntax des drop exit state definition Statements ist

drop [existing] exit state definition statename

Beschreibung

Beschreibung

Das drop exit state definition Statement wird benutzt um eine Exit State Definition zu löschen. Es wird als Fehler betrachtet wenn Exit State Profiles diese Exit State Definition immer noch benutzen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet wenn die spezifizierte Exit State Definition nicht existiert.

Ausgabe

Ausgabe

drop exit state mapping

Zweck

Das drop exist state mapping Statement wird eingesetzt um das spezifizierte Zweck Mapping zu löschen.

Syntax

Die Syntax des drop exit state mapping Statements ist

Syntax

drop [existing] exit state mapping mappingname

Beschreibung

Das drop exit state mapping Statement wird benutzt um Exit State Mappings zu löschen. Es wird als Fehler betrachtet, wenn Jobs oder Exit State Profiles immer noch dieses Exit State Mapping benutzen. Wenn das Schlüsselwort **existing** benutzt wird, wird es nicht als Fehler betrachtet, wenn das spezifizierte Exit State Mapping nicht existiert.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop exit state profile

drop exit state profile

Zweck

Zweck

Das drop exit state profile Statement wird eingesetzt um den spezifizierten Profile zu löschen.

Syntax

Syntax

Die Syntax des drop exit state profile Statements ist

drop [existing] exit state profile profilename

Beschreibung

Beschreibung

Das drop exit state profile Statement wird benutzt um eine Definition eines Exit State Profiles zu löschen. Es wird als Fehler betrachtet, wenn Jobs immer noch dieses Exit State Profile benutzen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Exit State Profile nicht existiert.

Ausgabe

Ausgabe

drop folder

User Commands

drop folder

Zweck

Das drop folder Statement wird eingesetzt um einen Folder und seinen Inhalt Zweck aus dem System zu entfernen.

Syntax

Die Syntax des drop folder Statements ist

Syntax

```
drop [ existing ] FOLDER_OR_JOB {, FOLDER_OR_JOB} [ cascade ] [ force ]
FOLDER_OR_JOB:
[ < folder folderpath | job definition folderpath > ]
```

Beschreibung

Mit dem drop folder Statement werden Folder und deren Inhalte aus dem System gelöscht. Es gibt zwei Optionen:

Cascade Mit der Cascade-Option werden Folder, Job Definitions und Subfolder gelöscht, allerdings nur wenn nicht an die Job Definitions referenziert wird z. B. als required Job.

Force Mit der force-Option werden Referenzen an Job Definitions ebenfalls entfernt. Force impliziert Cascade.

Folder können nicht gelöscht werden wenn sie nicht leer sind, es sei denn Cascade oder Force wird spezifiziert.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop footprint

drop footprint

Zweck

Zweck

Das drop footprint Statement wird eingesetzt um den spezifizierten Footprint zu löschen.

Syntax

Syntax

Die Syntax des drop footprint Statements ist

drop [existing] footprint footprintname

Beschreibung

Beschreibung

Das drop footprint Statement wird benutzt um Footprints und Resource Anforderungen zu löschen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Footprint nicht existiert.

Ausgabe

Ausgabe

drop group

User Commands

drop group

Zweck

Das drop group Statement wird eingesetzt um eine Gruppe aus dem System zu Zweck entfernen.

Syntax

Die Syntax des drop group Statements ist

Syntax

drop [existing] group groupname

Beschreibung

Das drop group Statement wird benutzt um eine Gruppe zu löschen. Wenn dort noch Gruppenmitglieder existieren, wird die Mitgliedschaft automatisch beendet. Es wird als Fehler betrachtet, wenn die Gruppe immer noch der Eigentümer eines Objektes ist. Beschreibung

Es ist nicht möglich eine Gruppe die als Default Gruppe eines Users definiert ist zu löschen

Wenn das **existing** Schlüsselwort benutzt wird, wird es nich als Fehler angesehen wenn die spezifizierte Gruppe nicht existiert.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop interval

drop interval

Zweck

Zweck

Das drop interval Statement wird eingesetzt um das spezifizierte Intervall zu löschen.

Syntax

Syntax

Die Syntax des drop interval Statements ist

drop [existing] interval intervalname

Beschreibung

Beschreibung

Das drop interval Statement wird benutzt um Intervalle zu löschen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Interval nicht existiert.

Ausgabe

Ausgabe

drop job definition

User Commands

drop job definition

Zweck

Das drop job definition Statement wird eingesetzt um das spezifizierte Schedul- Zweck ing Entity Objekt zu löschen.

Syntax

Die Syntax des drop job definition Statements ist

Syntax

drop [existing] job definition folderpath . jobname [force]

Beschreibung

Das drop job definition Statement löscht die angegebene Job Definition. Falls eine Job Definition referenziert wird (etwa als Required Job) kann sie nicht gelöscht werden, es sei denn man spezifiziert die Force-Option. Wird die Force-Option genutzt, werden alle Referenzen auf einer Job Definition ebenfalls gelöscht.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop named resource

drop named resource

Zweck

Zweck

Das drop named resource Statement wird eingesetzt um eine Klasse von Resourcen zu löschen.

Syntax

Syntax

Die Syntax des drop named resource Statements ist

drop [existing] named resource resourcepath [cascade]

Beschreibung

Beschreibung

Das drop named resource Statement wird benutzt um Named Resources zu löschen. Es wird als Fehler betrachtet, wenn die Named Resource immer noch in Scopes, Job Definitions und/oder Folder instanziiert ist und die **cascade** Option nicht spezifiziert ist.

Auf der anderen Seite werden Scope Resources, sowie Folder und Job Definition Resources gelöscht wenn die **cascade** Option spezifiziert ist.

Wenn irgendwelche Anforderungen, für die Named Resource die gelöscht werden sollen, existieren, schlägt das Statement fehl.

Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler angesehen wenn die spezifizierte Named Resource nicht existiert.

Ausgabe

Ausgabe

drop resource

User Commands

drop resource

Zweck

Das drop resource Statement wird eingesetzt um die Instanz einer Named Re- Zweck source von einem Scope, Folder oder Job Definition zu löschen.

Syntax

Die Syntax des drop resource Statements ist

Syntax

```
drop [ existing ] RESOURCE_URL [ force ]
```

RESOURCE_URL:

resource resourcepath in folderpath resource resourcepath in serverpath

Beschreibung

Das drop resource Statement wird benutzt um eine Resource zu löschen. Es wird als Fehler betrachtet wenn die Resource immer noch von Running Jobs allokiert ist. Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler angesehen, wenn die spezifizierte Resource nicht existiert.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop resource state definition

drop resource state definition

Zweck

Zweck

Das drop resource state definition Statement wird eingesetzt um die Definition zu löschen.

Syntax

Syntax

Die Syntax des drop resource state definition Statements ist

drop [existing] resource state definition statename

Beschreibung

Beschreibung

Das drop resource state definition Statement wird benutzt um Resource State Definitions zu löschen. Es wird als Fehler betrachtet, wenn Resource State Profiles immer noch diese Resource State Definition verwenden. Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler angesehen, wenn die spezifizierte Resource State Definition nicht existiert.

Ausgabe

Ausgabe

drop resource state mapping

Zweck

Das drop resource state mapping Statement wird eingesetzt um ein Mapping zu Zweck löschen.

Syntax

Die Syntax des drop resource state mapping Statements ist

Syntax

drop [existing] resource state mapping mappingname

Beschreibung

Das drop resource state mapping Statement wird benutzt um ein Resource State Mapping zu löschen. Es wird als Fehler betrachtet, wenn Job Definitions dieses Resource State Mapping benutzen. Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler gesehen, wenn das Resource State Mapping nicht existiert.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

drop resource state profile

drop resource state profile

Zweck

Zweck

Das drop resource state profile Statement wird eingesetzt um ein Resource State Profile zu löschen.

Syntax

Syntax

Die Syntax des drop resource state profile Statements ist

drop [existing] resource state profile profilename

Beschreibung

Beschreibung

Das drop resource state profile Statement wird benutzt um die Definition eines Resource State Profiles zu löschen. Es wird als Fehler betrachtet, wenn Named Resources immer noch dieses Resource State Profile benutzen. Wenn das existing Schlüsselwort benutzt wird, wird es nicht als Fehler betrachtet, wenn das spezifizierte Resource State Profile nicht existiert.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop schedule

User Commands

drop schedule

Zweck

Das drop schedule Statement wird eingesetzt um den spezifizierten Zeitplan zu Zweck löschen.

Syntax

Die Syntax des drop schedule Statements ist

Syntax

drop [existing] schedule schedulepath

Beschreibung

Das drop schedule Statement wird benutzt um Schedules zu löschen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Schedule nicht existiert.

Beschreibung

Wenn ein Schedule ein zugehöriges Scheduled Event hat, kann er nicht gelöscht werden. Löschen ist ebenfalls dann unmöglich, wenn Kindobjekte vorhanden sind.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

drop scheduled event

drop scheduled event

Zweck

Zweck

Der Zweck des drop scheduled event Statements ist, das Specified Scheduled Event zu löschen.

Syntax

Syntax

Die Syntax des drop scheduled event Statements ist

drop [existing] scheduled event schedulepath . eventname

Beschreibung

Beschreibung

Das drop interval Statement wird benutzt um Scheduled Events zu löschen. Wird das **existing** Schlüsselwort benutzt, wird es nicht als Fehler betrachtet, wenn das spezifizierte Schedule Event nicht existiert.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop scope

User Commands

drop scope

Zweck

Das drop scope Statement wird eingesetzt um einen Scope und seinen Inhalt aus Zweck der Scope-Hierarchie zu entfernen.

Syntax

Die Syntax des drop scope Statements ist

Syntax

```
\label{eq:cope} \textbf{drop} \ [\ \textbf{existing} \ ] < \textbf{scope} \ server path \ |\ \textbf{job server} \ server path > [\ \textbf{cascade} \ ]
```

Beschreibung

Dieses Statement ist synonym zu dem Drop Jobserver-Statement. Die Cascade- Beschreibung Option bewirkt, dass der Scope samt Inhalt gelöscht wird.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

drop trigger

drop trigger

Zweck

Zweck

Das drop trigger Statement wird eingesetzt um den spezifizierten Trigger zu löschen.

Syntax

Syntax

Die Syntax des drop trigger Statements ist

```
drop [ existing ] trigger triggername on <code>TRIGGEROBJECT</code> [ < noinverse | inverse > ]
```

TRIGGEROBJECT:

resource resourcepath in folderpath job definition folderpath object monitor objecttypename resource resourcepath in serverpath

Beschreibung

Beschreibung

Das drop trigger Statement wird benutzt um Trigger zu löschen. Ist das **existing** Schlüsselwort in Benutzung, wird es nicht als Fehler angesehen, wenn der spezifizierte Trigger nicht existiert.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

drop user

User Commands

drop user

Zweck

Das drop user Statement wird eingesetzt um den Benutzer aus dem System zu Zweck entfernen.

Syntax

Die Syntax des drop user Statements ist

Syntax

drop [existing] user username

Beschreibung

Das drop user Statement wird benutzt um einen User logisch zu löschen. Wenn das **existing** Schlüsselwort benutzt wird, wird es nicht als Fehler angesehen, wenn der spezifizierte User nicht existiert.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

10. finish commands

finish job

finish job

Zweck

Zweck

Der Zweck des finish job command ist, den Server über den Ablauf eines Jobs zu informieren.

Syntax

Syntax

Die Syntax des finish job Statements ist

```
finish job jobid
with exit code = signed_integer
finish job
```

with exit code = signed_integer

Beschreibung

Beschreibung

Das finish job-Kommando wird vom Jobserver genutzt um den Exit Code eines Prozesses dem Server zu melden. Im Rahmen von Reparaturarbeiten kann es auch für einen Administrator notwendig sein auf diese Weise das Terminieren eines Jobs dem Server mitzuteilen. Jobs können "sich selbst" fertig melden. Dazu verbinden sie sich mit dem Server und benutzten die zweite Form des Statements.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

11. get commands

get parameter

get parameter

Zweck

Zweck

Das get parameter Statement wird eingesetzt um den Wert des spezifizierten Parameters innerhalb des Kontext des anfordernden Jobs, entsprechend dem spezifizierten Jobs, zu bekommen.

Syntax

Syntax

Die Syntax des get parameter Statements ist

```
get parameter parametername [ < strict | warn | liberal > ]
get parameter of jobid parametername [ < strict | warn | liberal > ]
```

Beschreibung

Beschreibung

Das get parameter-Statement wird eingesetzt um den Wert des spezifierten Parameters innerhalb des Kontextes eines Jobs zu bekommen.

Die Zusatzoption hat dabei folgende Bedeutung:

Option	Bedeutung
strict	Der Server liefert einen Fehler, wenn der gefragte Parameter nicht ex-
	plizit in der Job Definition deklariert ist
warn	Es wird eine Meldung ins Logfile des Server geschrieben, wenn ver-
	sucht wird den Wert eines nicht deklarierten Parameters zu ermitteln.
liberal	Der Versuch nicht deklarierte Parameter ab zu fragen wird
	stillschweigend erlaubt.
D D (1. 1 1. 1

Das Defaultverhalten hängt von der Serverkonfiguration ab.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Feld	Beschreibung
VALUE	Wert des angeforderten Parameters

Tabelle 11.1.: Beschreibung der Output-Struktur des get parameter Statements

get submittag

Zweck

Das get submittag Statement wird eingesetzt um eine eindeutige Identifikation von dem Server zu bekommen. Diese Identifikation kann benutzt werden, um race conditions zwischen frontend und backend während des Submits zu verhindern.

Zweck

Syntax

Die Syntax des get submittag Statements ist

Syntax

get submittag

Beschreibung

Mit dem get submittag Statement bekommt man eine Identifikation von dem Server. Damit verhindert man race conditions zwischen frontend und backend wenn Jobs submitted werden.

Beschreibung

Eine solche Situation entsteht, wenn aufgrund eines Fehlers die Rückmeldung des Submits nicht ins Frontend eintrifft. Durch Benutzung eines Submittags kann das Frontend gefahrlos einen zweiten Versuch starten. Der Server erkennt ob der betreffende Job bereits submitted wurde und antwortet dementsprechend. Ein doppeltes Submitten des Jobs wird damit zuverlässig verhindert.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
VALUE	Das angeforderte Submit Tag

Tabelle 11.2.: Beschreibung der Output-Struktur des get submittag Statements

12. kill commands

kill session

kill session

Zweck

Zweck

Das Ziel der kill session ist, die spezifizierte session zu beenden.

Syntax

Syntax

Die Syntax des kill session Statements ist

kill session sid

Beschreibung

Beschreibung

Mittels list session Kommandos kann eine Liste von aktiven Sessions gezeigt werden. Die angezeigte Session Id kann benutzt werden um mittels des kill session Kommandos die betreffende Session zu terminieren. Nur Administratoren, das heisst Mitglieder der Gruppe Admin, dürfen dieses Statement benutzen. Es ist nicht möglich die eigene Session zu terminieren.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

13. link commands

link resource

link resource

Zweck

Zweck

Der Zweck des link resource Statements ist es in ein Scope eine Referenz auf eine Ressource aus einem anderen Scope zu bekommen.

Syntax

Syntax

Die Syntax des link resource Statements ist

link resource resourcepath in serverpath to < scope serverpath | job server serverpath > [force]

Beschreibung

Beschreibung

Mit dem link ressource Statement ist es möglich in einem Scope Ressourcen eines anderen Scopes sichtbar und benutzbar zu machen. Dies ist dann notwendig, wenn ein logischer Prozess Ressourcen aus mehr als einem Scope benötigt. Dies ist etwa bei Prozessen die mit einem Datenbanksystem kommunizieren durchaus der Fall. Aus Sicht des Systems kann ein Resource Link kaum von der Ressource auf die verwiesen wird unterschieden werden. Alle Operationen, wie etwa Allokieren, Sperren, das Lesen oder Setzen von Variablen erfolgen auf die Basisressource. Damit verhält sich der Link als wäre es die Basisressource. Der einzige Unterschied liegt in der Ansicht der Allocations. Bei der Basisressource werden alle Allocations gezeigt. Bei einem Link werden nur die Allocations gezeigt die über den Link erfolgen. Es ist ebenfalls möglich Links auf Links an zu legen.

Mit Hilfe der **force** option wird ein bereits vorhandener Link überschrieben. Eine bereits vorhandenen Ressource wird gelöscht und der Link wird angelegt. Natürlich sind diese Operationen nur dann möglich, wenn die Ressource bzw. Link nicht in benutzung ist, wenn also keine Allocations oder Reservierungen vorliegen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

14. list commands

list calendar

list calendar

Zweck

Zweck

Der Zweck des list calendar Statements ist es eine Übersicht über die anstehenden Jobs zu bekommen.

Syntax

Syntax

Die Syntax des list calendar Statements ist

```
list calendar [ with LC_WITHITEM {, LC_WITHITEM} ]
LC_WITHITEM:
     endtime = datetime
     filter = LC_FILTERTERM {or LC_FILTERTERM}
     starttime = datetime
     time zone = string
LC_FILTERTERM:
LC_FILTERITEM {and LC_FILTERITEM}
LC_FILTERITEM:
     ( LC_FILTERTERM {or LC_FILTERTERM} )
    job . identifier < cmpop | like | not like > RVALUE
     name like string
     not ( LC_FILTERTERM {or LC_FILTERTERM} )
RVALUE:
     expr ( string )
    number
     string
```

Beschreibung

Beschreibung

Mit dem list calendar Statement bekommt man eine Liste aller Kalendereinträge. die Liste ist sortiert nach Startdatum des Ausführungsobjektes.

Wenn eine Periode spezifiziert wird, werden auch solche Objekte angezeigt, deren Starzeit plus Expected Final Time in der selektierten Periode hineinfällt

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

188 l list calendar

list dependency definition

Zweck

Das list dependency definition Statement wird eingesetzt um eine Liste aller Zweck Abhängigkeiten einer Job Definition zu erstellen.

Syntax

Die Syntax des list dependency definition Statements ist

Syntax

list dependency definition folderpath

Beschreibung

Mit dem list dependency definition Statement bekommt man eine Liste aller Abhängigkeiten einer Job Definition.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Beschreibung

Beschreibung
Die Nummer des Repository Objektes
Der Folder in dem das abhängige Scheduling Entity liegt
Der Name des abhängigen Scheduling Entities
Der Folder in dem das benötigte Scheduling En-
tity liegt
Der Name des benötigten Scheduling Entities
Der Name des Objektes
Im Feld Unresolved Handling wird beschrieben was zu tun ist, wenn eine abhängige Objektinstanz im aktuellen Master Batch nicht vorhanden ist. Es gibt folgende Optionen: Ignore, Error und Suspend

list dependency definition

Fortsetzung der vorherigen Seite		
Feld	Beschreibung	
MODE	Der Dependency Mode gibt an in welchem Zusammenhang die Liste der Dependencies gesehen werden muss. Es gibt folgende Optio- nen: All und Any	
STATE_SELECTION	Die State Selection gibt an, wie die benötigten Exit States ermittelt werden. Es gibt die Optio- nen FINAL, ALL_REACHABLE, UNREACH- ABLE und DEFAULT. Im Falle von FINAL kön- nen die benötigte Exit States expliziert aufge- führt sein	
ALL_FINALS	Dieses Feld gibt an ob die Abhängigkeit breits bei erreichen eines Final State erfült ist (True) oder die benötigten States explizit aufgeführt sind (False)	
CONDITION	Im Feld Condition wird die Bedingung die erfült werden muss eingetragen	
STATES	Hier steht die Liste von allen gültigen Exit States, welche das benötigte Objekt haben muss, damit die Abhängigkeit erfüllt wird und der ab- hängige Job starten kann	

Tabelle 14.1.: Beschreibung der Output-Struktur des list dependency definition Statements

list dependency hierarchy

Zweck

Das list dependency hierarchy Statement wird eingesetzt um eine Liste aller Ab- Zweck hängigkeiten eines Submitted Entities zu bekommen.

Syntax

Die Syntax des list dependency hierarchy Statements ist

Syntax

list dependency hierarchy *jobid* [with EXPAND]

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Mit dem list dependency hierarchy Statement bekommt man eine Liste aller Abhängigkeiten eines Submitted Entities.

Beschreibung

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Id der Dependency Instance
DD_ID	Die ID der Dependency Definition
DEPENDENT_ID	Hierbei handelt es sich um die ID des abhängi-
	gen Jobs
Fortsetzung auf der nächsten	Seite

list dependency hierarchy

Feld	Beschreibung
DEPENDENT_NAME	Das ist der vollqualifizierte Name des abhängi gen Jobs
REQUIRED_ID	Hierbei handelt es sich um die ID der benötigten Jobs
REQUIRED_NAME	Hierbei handelt es sich um den vollquali fizierten Namen des benötigten Jobs
DEP_STATE	Hierbei handelt es sich um den aktuellen Status der Abhängigkeitsbeziehung. Es gibt folgende Ausprägungen: Open, Fullfilled und Filed
DEPENDENCY_PATH	Hierbei handelt es sich um eine durch ';' ge trennte Liste von Job Hierarchies (Parent-Child Beziehungen). Jede Job Hierarchie ist eine Liste von Pfadnamen jeweils durch ':' getrennt
SE_DEPENDENT_ID	Die ID des abhängigen Scheduling Entities
SE_DEPENDENT_NAME	Der vollqualifizierte Name des Abhängiger Scheduling Entities
SE_REQUIRED_ID	Die ID des benötigten Scheduling Entities
SE_REQUIRED_NAME	Der vollqualifizierte Name des benötigter Scheduling Entities
DD_NAME	Das ist der Name der Dependency Definition
UNRESOLVED_HANDLING	Dieses Feld zeigt die Einstellung des Feldes Or Unresolved Error aus der Submit Maske. Es gib folgende Optionen: Ignore, Suspend und None
MODE	Gibt den aktuell verwendeten Dependency Mode an (ALL_FINAL oder JOB_FINAL)
STATE_SELECTION	Die State Selection gibt an, wie die benötigter Exit States ermittelt werden. Es gibt die Optio nen FINAL, ALL_REACHABLE, UNREACH ABLE und DEFAULT. Im Falle von FINAL kön nen die benötigte Exit States expliziert aufge führt sein
MASTER_ID	Hierbei handelt es sich um die ID des Mas ter Jobs, welcher submitted wurde, um diese Laufzeitobjekt zu erzeugen
SE_TYPE	Hierbei handelt es sich um den Typ de Scheduling Entities (Job, Batch oder Milestone)
PARENT_ID	Hierbei handelt es sich um die ID des Parent Laufzeitobjektes welche den aktuellen Job sub mitted hat. Hat der Job kein Parent, wird NONI angezeigt

Fortsetzung der vorherigen Sei	ite
Feld	Beschreibung
PARENT_NAME	Hierbei handelt es sich um den vollquali- fizierten Namen des Parent-Laufzeitobjektes welche den aktuellen Job submitted hat
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCOPE	Hierbei handelt es sich um den vollquali- fizierten Namen des Jobservers auf dem der Job gestartet wurde. Wurde der Job noch nicht ges- tartet, wird 'null' angezeigt
EXIT_CODE	Beim Exit Code handelt es sich um den Exit Wert, den das Run Programm bei der Beendi- gung des Prozesses hatte
PID	Das ist die Prozess ID des Job Executors
EXTPID	Das ist die ID des Prozesses der ausgeführt wird
JOB_STATE	Der aktuelle Job State
JOB_ESD	Hierbei handelt es sich um den Exit Status des Jobs. Wenn der Job noch nicht beendet ist, wird 'null' angezeigt
FINAL_ESD	Hierbei handelt es sich um den Merged Exit Status
JOB_IS_FINAL	Gibt an ob der Job Final ist (True) oder nicht (False)
CNT_REQUIRED	Die Anzahl der Jobs die vom aktuellen Job abhängen, wenn der Job im Status dependency_wait ist
CNT_RESTARTABLE	Die Anzahl der Child Jobs im Status restartable
CNT_SUBMITTED	Die Anzahl der Child Jobs im Status submitted
CNT_DEPENDENCY_WAIT	Die Anzahl der Child Jobs im Status dependen- cy_wait
CNT_RESOURCE_WAIT	Die Anzahl der Child Jobs im Status resource_wait
CNT_RUNNABLE	Die Anzahl der Child Jobs im Status runnable
CNT_STARTING	Die Anzahl der Child Jobs im Status starting
CNT_STARTED	Die Anzahl der Child Jobs im Status started
CNT_RUNNING	Die Anzahl der Child Jobs im Status running
CNT_TO_KILL	Die Anzahl der Child Jobs im Status to_kill
CNT_KILLED	Die Anzahl der Child Jobs im Status killed
CNT_CANCELLED	Die Anzahl der Child Jobs im Status cancelled
CNT_FINAL	Die Anzahl der Child Jobs im Status final
Fortsetzung auf der nächsten S	eite

list dependency hierarchy

Feld	te Beschreibung
CNT_BROKEN_ACTIVE	Die Anzahl der Child Jobs im Status bro-
CN1_DROREN_ACTIVE	ken active
CNT_BROKEN_FINISHED	Die Anzahl der Child Jobs im Status bro-
	ken_finished
CNT_ERROR	Die Anzahl der Child Jobs im Status error
CNT_SYNCHRONIZE_WAIT	Die Anzahl der Child Jobs im Status synchro-
	nize_wait
CNT_FINISHED	Die Anzahl der Child Jobs im Status finished
SUBMIT_TS	Der Zeitpunkt an dem der Job submitted wurde
SYNC_TS	Der Zeitpunkt an dem der Job den Status Syn-
	chronize_Wait erreicht hat
RESOURCE_TS	
RUNNABLE_TS	Der Zeitpunkt an dem der Job den Status
	Runnable erreicht hat
START_TS	Der Zeitpunkt an dem der Job gestartet wurde
FINSH_TS	
FINAL_TS	
ERROR_MSG	Die Fehlermeldung die beim Erreichen des Sta-
	tus Error ausgegeben wurde
DEPENDENT_ID_ORIG	Die Id des Objektes der die Abhängigkeit definiert hat
DEPENDENCY OPERATION	
DEPENDENCY_OPERATION	Die Dependency Operation gibt an ob alle Abhängigkeiten (All) oder nur eine einzige Abhängigkeiten
	hängigkeit erfüllt sein muss
CHILD_TAG	Marker zur Unterscheidung mehrerer dynamic
51122_1116	submitted Children
CHILDREN	Die Anzahl der Children des Jobs
REQUIRED	Die Anzahl der abhängigen Jobs
DD_STATES	Eine kommagetrennte Liste der benötigten Exit
	States
IS_SUSPENDED	Dieses Feld gibt an ob der Job suspended (True)
	ist oder nicht (False)
PARENT_SUSPENDED	Dieses Feld gibt an ob der Job über einen seiner
	Parents suspended ist (True) oder nicht (False)
CNT_UNREACHABLE	Die Anzahl Children deren Dependencies nicht
	erfüllt werden können
DEPENDENT_PATH_ORIG	Der vollqualifizierte Name des Objektes der die
	Abhängigkeit definiert hat

list dependency hierarchy

User Commands

Fortsetzung der vorher	igen Seite
Feld	Beschreibung
IGNORE	Ignore gibt an, ob diese Abhängigkeit ignoriert wird (True) oder nicht (False)

Tabelle 14.2.: Beschreibung der Output-Struktur des list dependency hierarchy Statements

list environment

list environment

Zweck

Zweck

Das list environment Statement wird eingesetzt um eine Liste von definierten Environments zu bekommen.

Syntax

Syntax

Die Syntax des list environment Statements ist

list environment

Beschreibung

Beschreibung

Das list environment Statement wird benutzt um eine Liste von definierten Environments die für den Benutzer sichtbar sind zu bekommen.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Environments
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.3.: Beschreibung der Output-Struktur des list environment Statements

list event User Commands

list event

Zweck

Das list event Statement wird eingesetzt um eine Liste von allen definierten Zweck Events zu bekommen.

Syntax

Die Syntax des list event Statements ist

Syntax

list event

Beschreibung

Das list event Statement erzeugt eine Liste aller definierten Events.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCHEDULING_ENTITY	Batch oder Job der submitted wird wenn dieses
	Event eintritt
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.4.: Beschreibung der Output-Struktur des list event Statements

list event | 197

list exit state definition

list exit state definition

Zweck

Zweck

Das list exit state definition Statement wird eingesetzt um eine Liste aller definierten Exit States zu bekommen.

Syntax

Syntax

Die Syntax des list exit state definition Statements ist

list exit state definition

Beschreibung

Beschreibung

Mit dem list exit state definition Statement bekommt man eine Liste aller Exit States.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.5.: Beschreibung der Output-Struktur des list exit state definition Statements

list exit state mapping

Zweck

Das list exit state mapping Statement wird eingesetzt um eine Liste aller defin- Zweck erten Mappings zu bekommen.

Syntax

Die Syntax des list exit state mapping Statements ist

Syntax

list exit state mapping

Beschreibung

Mit dem list exit state mapping Statement bekommt man eine Liste aller definierten *Beschreibung* Mappings.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.6.: Beschreibung der Output-Struktur des list exit state mapping Statements

list exit state profile

list exit state profile

Zweck

Zweck

Das list exit state profile Statement wird eingesetzt um eine Liste von allen definierten Exit State Profiles zu bekommen.

Syntax

Syntax

Die Syntax des list exit state profile Statements ist

list exit state profile

Beschreibung

Beschreibung

Mit dem list exit state profile Statement bekommt man eine Liste aller definierten Exit State Profiles.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
DEFAULT_ESM_NAME	Default Exit State Mapping ist aktiv wenn der Job selbst nichts anderes angibt
IS_VALID	Flag Anzeige über die Gültigkeit dieses Exit State Profiles
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.7.: Beschreibung der Output-Struktur des list exit state profile Statements

list folder

User Commands

list folder

Zweck

Das list folder Statement wird eingesetzt um eine Liste mit allen Folder die in Zweck dem System definiert sind zu bekommen.

Syntax

Die Syntax des list folder Statements ist

Syntax

```
list folder folderpath [ with WITHITEM {, WITHITEM} ]

WITHITEM:
    expand = none
    expand = < ( id {, id} ) | all >
        | FILTERTERM {or FILTERTERM}

FILTERITEM:
    fILTERITEM {and FILTERITEM}

FILTERITEM:
        ( FILTERTERM {or FILTERTERM} )
        | name like string
        | not ( FILTERTERM {or FILTERTERM} )
        | owner in ( groupname { , groupname} )
```

Beschreibung

Mit dem list folder Statement bekommt man eine Liste des angegebenen Folders Beschreibung mit allen direkten Childfolders.

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

filter Die Childfolders können nach ihrem Namen selektiert werden. Für die genaue Syntax der Regular Expressions sei auf die offizielle Java Dokumentation verwiesen. Die verschiedene Bedingungen können mittels **and** und **or** miteinander kombiniert werden. Dabei gilt die übliche Auswertungsreihenfolge der Operatoren (**and** vor **or**).

list folder | 201

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

list folder

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
ТҮРЕ	Der Type gibt die Art des Objektes an. Es gibt folgende Optionen: Batch, Milestone, Job und Folder
RUN_PROGRAM	Im Feld Run Program kann eine Kommandozeile angegeben werden, die das Skript oder Programm startet
RERUN_PROGRAM	Das Feld Rerun Program gibt das Kommando an, welches bei einer wiederholten Ausführung des Jobs nach einem Fehlerzustand (Rerun) aus- geführt werden soll
KILL_PROGRAM	Das Kill Program bestimmt welches Programm ausgeführt werden soll, um einen aktuell laufenden Job zu beenden
WORKDIR	Hierbei handelt es sich um das Working Directory des aktuellen Jobs
LOGFILE	Das Feld Logfile gibt an in welche Datei alle nor- malen Ausgaben des Run Programs ausgegeben werden sollen. Unter normalen Ausgaben sind alle Ausgaben gemeint die den normalen Aus- gabekanal (STDOUT unter Unix) benutzen
TRUNC_LOG	Gibt an ob das Logfile erneuert werden soll oder nicht
ERRLOGFILE	Das Feld Error Logfile gibt an, in welcher Datei alle Fehlerausgaben des Run Programs aus- gegeben werden sollen
TRUNC_ERRLOG	Gibt an ob das Error Logfile erneuert werden soll oder nicht
Fortsetzung auf der nächsten S	eite

list folder User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
EXPECTED_RUNTIME	Die Expected Runtime beschreibt die erwartete Zeit die ein Job für seine Ausführung benötigt
EXPECTED_FINALTIME	Die Expected Finaltime beschreibt die erwartete Zeit die ein Job oder Batch samt Children für seine Ausführung benötigt
GET_EXPECTED_RUNTIME	Hierbei handelt es sich um ein reserviertes Feld für zukünftige Erweiterungen
PRIORITY	Das Feld Priority gibt an mit welcher Dringlichkeit ein Prozess, falls er gestartet werden soll, vom Scheduling System berück- sichtigt wird
MIN_PRIORITY	Minimale effektive Priorität die durch das natürliche Altern erreicht werden kann
AGING_AMOUNT	Die Anzahl Zeiteinheiten nach der die effektive Priorität um 1 erhöht wird
AGING_BASE	Die Zeiteinheit die für das Alterungsintervall genutzt wird
SUBMIT_SUSPENDED	Flag das angibt ob das Objekt nach dem Submit suspended werden soll
MASTER_SUBMITTABLE	Der Job der durch den Trigger gestartet wird, wird als eigener Master Job submitted und hat keinen Einfluss auf den aktuellen Master Job- Lauf des triggernden Jobs
SAME_NODE	Obsolete
GANG_SCHEDULE	Obsolete
DEPENDENCY_MODE	Der Dependency Mode gibt an in welchem Zusammenhang die Liste der Dependencies gesehen werden muss. Es gibt folgende Optio- nen: All und Any
ESP_NAME	Hierbei handelt es sich um den Namen des Exit State Profiles
ESM_NAME	Hierbei handelt es sich um den Namen des Exit State Mappings
ENV_NAME	Hierbei handelt es sich um den Namen des Environments
FP_NAME	Hierbei handelt es sich um den Namen des Footprints
Fortsetzung auf der nächsten Seite	

User Commands list folder

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
SUBFOLDERS	Hierbei handelt es sich um die Anzahl Folder unterhalb des Folders
ENTITIES	Hierbei handelt es sich um die Anzahl Jobs und Batches unterhalb des Folders
HAS_MSE	In dem Folder befindet sich mindestens ein Job der als Master Submittable ausgeführt werden kann
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
IDPATH	Id Pfad zum Objekt
HIT	Zeile ist ein suchtreffer Y/N

Tabelle 14.8.: Beschreibung der Output-Struktur des list folder Statements

list footprint

User Commands

list footprint

Zweck

Das list footprint Statement wird eingesetzt um eine Liste aller definierten Foot- Zweck prints zu bekommen.

Syntax

Die Syntax des list footprint Statements ist

Syntax

list footprint

Beschreibung

Mit dem list footprint Statement bekommt man eine Liste aller definierten Footprints.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.9.: Beschreibung der Output-Struktur des list footprint Statements

list group

list group

Zweck

Zweck

Das list group Statement wird eingesetzt um eine Liste von allen definierten Gruppen zu bekommen.

Syntax

Syntax

Die Syntax des list group Statements ist

list group

Beschreibung

Beschreibung

Mit dem list group Statement bekommt man eine Liste aller definierten Gruppen.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.10.: Beschreibung der Output-Struktur des list group Statements

list interval

User Commands

list interval

Zweck

Das list interval Statement wird eingesetzt um eine Liste aller definierten Inter- Zweck valle zu bekommen.

Syntax

Die Syntax des list interval Statements ist

Syntax

list interval

Beschreibung

Mit dem list interval Statement bekommt man eine Liste aller definierten Intervalle.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
STARTTIME	Der Anfang des Intervals. Vor dieser Zeit werden keine Flanken generiert.
ENDTIME	Das Ende des Intervals. Nach dieser Zeit werden keine Flanken generiert.
BASE	Die Periode des Intervals.
DURATION	Die Dauer eines Blocks.
SYNCTIME	Die Zeit mit der das Interval synchronisiert wird. Die erste Periode des Intervals startet zu dieser Zeit.
INVERSE	Die Angabe ob die Auswahlliste positiv oder negativ aufgefasst werden soll.

User Commands list interval

Fortsetzung der vorherigen Seite		
Feld	Beschreibung	
EMBEDDED	Das Interval aus dem nachträglich eine Auswahl getroffen wird.	
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält	

Tabelle 14.11.: Beschreibung der Output-Struktur des list interval Statements

list job

User Commands

list job

Zweck

Das list job Statement wird eingesetzt um eine Liste von Submitted Entities zu Zweck bekommen, basierend auf das spezifizierte Selektionskriterium.

Syntax

Die Syntax des list job Statements ist

Syntax

```
list job [ jobid {, jobid} ] [ with WITHITEM {, WITHITEM} ]
WITHITEM:
     expand = none
    expand = < ( id {, id} ) | all >
     FILTERTERM {or FILTERTERM}
   | mode = < <u>list</u> | tree >
     parameter = ( parametername {, parametername} )
FILTERTERM:
FILTERITEM {and FILTERITEM}
FILTERITEM:
     ( FILTERTERM {or FILTERTERM} )
     < final | restartable | pending >
     exit state in ( statename {, statename} )
     < history | future > = period
     history between period and period
     job . identifier < cmpop | like | not like > RVALUE
     job in ( jobid {, jobid} )
     job server in ( serverpath {, serverpath} )
     job status in ( JOBSTATE {, JOBSTATE} )
     master
     master_id in ( jobid {, jobid} )
     merged exit state in ( statename {, statename} )
     name in ( folderpath {, folderpath} )
     name like string
     node in ( nodename {, nodename} )
     not ( FILTERTERM {or FILTERTERM} )
     owner in ( groupname {, groupname} )
     submitting user in ( groupname {, groupname} )
     warning
```

list job

RVALUE:

expr (string)
| number
| string

JOBSTATE:

broken active broken finished cancelled

dependency wait

error final

finished

killed

resource wait

runnable

running

started

starting

submitted

synchronize wait

to kill

unreachable

Beschreibung

Beschreibung

Mit dem list job Statement bekommt man eine Liste von Submitted Entities. Die Auswahl der Jobs kann duch Angabe eines Filters beliebig fein spezifiziert werden. Zudem können Jobparameternamen spezifiziert werden die darauf in der Ausgabe sichtbar werden.

Das Statement list job ohne weitere Angaben ist gleichbedeutend mit dem Statement list job with master und gibt also die Liste aller Masterjobs und -batches aus.

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

mode Im Mode **list** wird einfach eine Liste von selektierten Jobs ausgegeben. Wird dagegen **tree** als Mode angegeben, werden zu jedem selektierten Job alle Parents ausgegeben.

parameter Durch Angabe von Parameternamen kann zusätzliche Information zu den selektierten Jobs ausgegeben werden. Die Parameter werden im jeweiligen Jobkontext ausgewertet und der Wert des Parameters wird in der Ausgabe sichtbar gemacht. Falls dies fehl schlägt, wird ein Leerstring ausgegeben. Das heisst, dass die Angabe von nicht existierende Parameternamen keine negative Folgen hat. Auf diese Weise können Status- oder Fortschrittinformationen von Jobs leicht übersichtlich dargestellt werden.

filter Zur Filterung aller im System vorhandenen Jobs kann von einer Vielzahl an Filter gebrauch gemacht werden. Die einzelne Filter können mittel boolsche Operatoren miteinander kombiniert werden. Dabei gilt die übliche Prioritätsreihenfolge der Operatoren.

Im Folgenden werden die einzelne Filtermöglichkeiten kurz beschrieben.

FINAL, RESTARTABLE, PENDING Dieser Filter selektiert alle Jobs die final bzw. restartable oder pending sind.

EXIT STATE Alle Jobs die einen Exit State haben die in der spezifizierte Liste vorkommt, werden selektiert. Es handelt sich hier um den jobeigene Exit State, nicht den merged Exit State, der auch die Exit States der Kinder berücksichtigt.

<u>HISTORY</u> Durch Angabe einer History werden nur die Jobs selektiert die frühestens vor der angegebene Zeit final geworden sind. Nonfinal Jobs werden alle selektiert.

<u>FUTURE</u> Durch Angabe einer Future werden ebenfalls geplante zukünftige Jobs ausgegeben. Diese Ereignisse werden auf Basis von Scheduled Events sowie Calendar-Einträgen ermittelt. Als Status solcher Jobs wird "SCHEDULED" ausgegeben.

JOB.IDENTIFIER Mittels dieses Filters werden alle die Jobs selektiert, deren angegebene Parameter der Bedingung erfüllt. Auf diese Weise können etwa alle Jobs eines Entwicklers leich selektiert werden. (Unter der Annahme, dass jeder Job einen Parameter mit dem Enwicklername hat, natürlich).

Mit Hilfe der **expr** Funktion können auch Berechnungen durchgeführt werden. Der Ausdruck

```
job.starttime < expr('job.sysdate - job.expruntime * 1.5')</pre>
```

ermittelt die Jobs, die ihre zu erwartenden Laufzeit um mehr als 50% überschritten haben.

JOB IN (ID, . . .) Diese Filteroption ist gleichbedeutend mit der Angabe von Jobids nach "list job". Nur die Jobs mit einer der angegebenen Ids werden selektiert.

JOB SERVER Nur die Jobs die auf den angegebenen Jobserver laufen werden selektiert.

<u>JOB STATUS</u> Dieser Filter selektiert nur die Jobs die einer der angegebenen Jobstatus haben. Es ist z.B. leicht alle Jobs im Status **broken_finished** zu finden.

MASTER Nur die Masterjobs und -batches werden selektiert.

MASTER_ID Nur Jobs die zu den spezifizierten Masterjobs und -batches gehören werden selektiert.

list job

MERGED EXIT STATE Alle Jobs die einen merged Exit State haben die in der spezifizierte Liste vorkommt, werden selektiert. Es handelt sich hier um den Exit State der resultiert aus dem eigenen Exit State in Kombination mit den Exit States der Kinder.

NAME IN (FOLDERPATH, ...) Die Jobs deren zugehöriger Scheduling Entity in der spezifizierten Liste vorkommt, werden selektiert.

NAME LIKE STRING Die Jobs deren zugehöriger Scheduling Entity den passenden Namen hat, werden selektiert. Für nähere Information bezüglich der Syntax von Regular Expressions sei auf die offizielle Java Dokumentation verwiesen.

<u>NODE</u> Jobs die auf einem der spezifizierten Nodes laufen werden selektiert. In diesem Kontext bezeichnet der Node den Eintrag für **node** des Jobservers.

OWNER Nur die Jobs der angegebenen Owners (Gruppen) werden selektiert.

<u>SUBMITTING USER</u> Nur die Jobs die vom angegebenen Benutzer submitted wurden, werden selektiert.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
MASTER_ID	Hierbei handelt es sich um die ID des Master Jobs
HIERARCHY_PATH	Der Hierarchy Path stellt den kompletten Pfad des aktuellen Eintrages dar. Die einzelnen Hi- erarchiestufen werden mittels eines Punktes ge- trennt
SE_TYPE	Hierbei handelt es sich um den Typ Scheduling Entities
PARENT_ID	Hierbei handelt es sich um die ID des Parents
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCOPE	Der Scope, bzw. Jobserver, dem der Job zuge- ordnet ist
HTTPHOST	Der Hostname des Scopes für den Zugriff auf Logfiles via HTTP
Fortsetzung auf der nächsten Seite	

list job User Commands

Fortsetzung der vorherigen Sei Feld	Beschreibung
HTTPPORT	Die HTTP Portnummer des Jobservers für der
IIIIIIORI	Zugriff auf Logfiles via HTTP
EXIT_CODE	Der Exit Code des ausgeführten Prozesses
PID	Bei der Pid handelt es sich um die Prozessi
	dentifikationsnummer des überwachenden Job serverprozesses auf dem jeweiligen Hostsysten
EXTPID	Die Extpid ist die Prozessidentifikationsnum mer des Nutzprozesses
STATE	Der State ist der aktuelle Status des Jobs
IS_DISABLED	Zeigt an ob der Job bzw. Batch disabled submit
_	ted wurde.
IS_CANCELLED	Zeigt an ob ein Cancel auf den Job ausgeführ wurde
JOB_ESD	Der job_esd ist der Exit State des Jobs
FINAL_ESD	Der final_esd ist der zusammengefasste Exi State vom Job oder Batch mit allen Child Exi States
JOB_IS_FINAL	Dieses Feld gibt an ob der Job selbst final ist
CNT_RESTARTABLE	Hierbei handelt es sich um die Children des Job die sich in dem Status Restartable befinden
CNT_SUBMITTED	Die Anzahl der Child Jobs mit dem Submitted Status
CNT_DEPENDENCY_WAIT	Die Anzahl der Child Jobs mit dem Dependency Wait Status
CNT_SYNCHRONIZE_WAIT	Die Anzahl der Child Jobs mit dem Synchroniz Wait Status
CNT_RESOURCE_WAIT	Die Anzahl der Child Jobs mit dem Resourc Wait Status
CNT_RUNNABLE	Die Anzahl der Child Jobs im Status runnable
CNT_STARTING	Die Anzahl der Child Jobs im Status starting
CNT_STARTED	Die Anzahl der Child Jobs im Status started
CNT_RUNNING	Die Anzahl der Child Jobs im Status running
CNT_TO_KILL	Die Anzahl der Child Jobs im Status to_kill
CNT_KILLED	Die Anzahl der Child Jobs im Status killed
CNT_CANCELLED	Die Anzahl der Child Jobs im Status cancelled
CNT_FINISHED	Die Anzahl der Child Jobs im Status finished
CNT_FINAL	Die Anzahl der Child Jobs im Status final
CNT_BROKEN_ACTIVE	Die Anzahl der Child Jobs im Status broken_active

list job | 213

list job

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CNT_BROKEN_FINISHED	Die Anzahl der Child Jobs im Status bro- ken_finished
CNT_ERROR	Die Anzahl der Child Jobs im Status error
CNT_UNREACHABLE	Die Anzahl der Child Jobs die Unreachable geworden sind
CNT_WARN	Die Anzahl der Child Jobs für die eine Warnung vorliegt
SUBMIT_TS	Hierbei handelt es sich um den Zeitpunkt zu dem der Job submitted wird
RESUME_TS	Der Zeitpunkt zu dem der Job automatisch resumed wird
SYNC_TS	Der Zeitpunkt zu dem der Job in den Status syn- chronize_wait gewechselt ist
RESOURCE_TS	Der Zeitpunkt zu dem der Job in den Sttaus re- source_wait gewechselt ist
RUNNABLE_TS	Der Zeitpunkt zu dem der Job in den Sttaus runnable gewechselt ist
START_TS	Der Zeitpunkt zu dem der Job vom Jobserver als gestarted gemeldet wurde
FINISH_TS	Hierbei handelt es sich um den Zeitpunkt in dem der Job beendet wird
FINAL_TS	Der Zeitpunkt zu dem der Job einen final State erreichte
PRIORITY	Die statische Priorität eines Jobs. Diese setzt sich zusammen aus der definierten Priorität und den Nice-Values der Parent(s)
DYNAMIC_PRIORITY	Die dynamische Priorität des Jobs. Diese ist die abhängig von der Wartezeit korrigierte statische Priorität
NICEVALUE	Die Nice Value ist die Korrektur der Priorität der Kinder
MIN_PRIORITY	Diese ist der minimale Wert der dynamische priorität
AGING_AMOUNT	Der Aging Amount gibt an nach wieviele Zeit- einheiten die dynamische Priorität eines Jobs um einen Punkt hochgesetzt wird

Fortsetzung auf der nächsten Seite

list job User Commands

Feld	Beschreibung
AGING_BASE	Die Aging Base gibt an um welche Zeiteinhei-
	es beim Aging Amount geht
ERROR_MSG	Die Fehlermeldung die beschreibt warum der
	Job in den Status error gewechselt ist
CHILDREN	Die Anzahl Children des Jobs oder Batches
HIT	Dieses Feld gibt an ob der Job aufgrund Fiterkri-
	terien ausgewählt wurde, oder nicht
HITPATH	Dieses Feld gibt an, dass der Job ein direkter
	oder indirekter Parent eines selektierten Jobs is
SUBMITPATH	Dies ist die Liste der submitting Parents. Im
	Gegensatz zu der allgemeine Parent-Child Hi-
	erarchie, ist diese immer eindeutig
IS_SUSPENDED	Dieses Feld gibt an ob der Job oder Batch selbs
	suspended ist
IS_RESTARTABLE	Dieses Feld gibt an ob der Job restartable ist
PARENT_SUSPENDED	Dieses Feld gibt an ob der Job oder Batch über
	einser seiner Parents suspended ist
CHILDTAG	Der Tag womit mehrere Children von einander
	unterschieden werden können
IS_REPLACED	Dieses Feld gibt an ob der Job oder Batch durch
	einen anderen ersetzt wurde
WARN_COUNT	Dies ist die Anzahl unbehandelte Warnings
CHILD_SUSPENDED	Die Anzahl der Children die Suspended wur-
	den
CNT_PENDING	Die Anzahl der Children in einem Pending Sta-
	tus
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält
WORKDIR	Name des Working Directorys des
	Nutzprozesses
LOGFILE	Name des Logfiles des Nutzprozesses. Hier
	werden die Ausgaben nach stdout protokol-
EDDI OCELLE	liert
ERRLOGFILE	Name des Error Logfiles des Nutzprozesses
	Hier werden die Ausgaben nach stderr pro- tokolliert
Fortsetzung auf der nächste	

User Commands list job

Fortsetzung der vorherigen Seite

Feld Beschreibung

Tabelle 14.12.: Beschreibung der Output-Struktur des list job Statements

list job definition hierarchy

Zweck

Das list job definition hierarchy Statement wird eingesetzt um den kompletten Zweck Jobtree des spezifizierten Jobs zu bekommen.

Syntax

Die Syntax des list job definition hierarchy Statements ist

Syntax

list job definition hierarchy *folderpath* [with EXPAND]

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Mit dem list job definition hierarchy Statement bekommt man die komplette Beschreibung Baumstruktur des spezifizierten Jobs.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
ТҮРЕ	Der Type gibt die Art des Objektes an. Es gibt folgende Optionen: Batch, Milestone, Job und Folder
RUN_PROGRAM	Im Feld Run Program kann eine Komman- dozeile angegeben werden, die das Skript oder Programm startet
Fortsetzung auf der nächsten Seite	

list job definition hierarchy

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
RERUN_PROGRAM	Das Feld Rerun Program gibt das Kommando an, welches bei einer wiederholten Ausführung des Jobs nach einem Fehlerzustand (Rerun) aus- geführt werden soll
KILL_PROGRAM	Das Kill Program bestimmt welches Programm ausgeführt werden soll, um einen aktuell laufenden Job zu beenden
WORKDIR	Hierbei handelt es sich um das Working Directory des aktuellen Jobs
LOGFILE	Das Feld Logfile gibt an in welche Datei alle nor- malen Ausgaben des Run Programs ausgegeben werden sollen. Unter normalen Ausgaben sind alle Ausgaben gemeint die den normalen Aus- gabekanal (STDOUT unter Unix) benutzen
TRUNC_LOG	Gibt an ob das Logfile erneuert werden soll oder nicht
ERRLOGFILE	Das Feld Error Logfile gibt an, in welcher Datei alle Fehlerausgaben des Run Programs aus- gegeben werden sollen
TRUNC_ERRLOG	Gibt an ob das Error Logfile erneuert werden soll oder nicht
EXPECTED_RUNTIME	Die Expected Runtime beschreibt die erwartete Zeit die ein Job für seine Ausführung benötigt
GET_EXPECTED_RUNTIME	Hierbei handelt es sich um ein reserviertes Feld für zukünftige Erweiterungen
PRIORITY	Das Feld Priority gibt an mit welcher Dringlichkeit ein Prozess, falls er gestartet werden soll, vom Scheduling System berück- sichtigt wird
SUBMIT_SUSPENDED	Der Parameter Submit Suspended gibt an in welcher Form das Child Objekt beim Start verzögert wird oder aber sofort gestartet wer- den kann. Es gibt folgende Optionen: Yes, No und Childsuspend
Fortsetzung auf der nächsten Seite	

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
MASTER_SUBMITTABLE	Der Job der durch den Trigger gestartet wird,
	wird als eigener Master Job submitted und hat
	keinen Einfluss auf den aktuellen Master Job-
	Lauf des triggernden Jobs
SAME_NODE	Obsolete
GANG_SCHEDULE	Obsolete
DEPENDENCY_MODE	Der Dependency Mode gibt an in welchem Zusammenhang die Liste der Dependencies gesehen werden muss. Es gibt folgende Optionen: All und Any
ESP_NAME	Hierbei handelt es sich um den Namen des Exit State Profiles
ESM_NAME	Hierbei handelt es sich um den Namen des Exit
_	State Mappings
ENV_NAME	Hierbei handelt es sich um den Namen des En-
	vironments
FP_NAME	Hierbei handelt es sich um den Namen des
	Footprints
CHILDREN	Hierbei handelt es sich um die Anzahl direkter Children
SH_ID	Die Id der Hierarchy Definition
IS_STATIC	Flag das statische oder dynamische Submits von diesem Job anzeigt
IS_DISABLED	Flag das angibt ob das Child ausgeführt oder übersprungen wird
SH_PRIORITY	Das Feld Priority gibt an mit welcher Dringlichkeit ein Prozess, falls er gestartet werden soll, vom Scheduling System berück- sichtigt wird
SH_SUSPEND	Der Schalter Submit Suspended gibt die Möglichkeit den tatsächlichen Start eines Ablaufes zu verzögern
SH_ALIAS_NAME	Mit dem Alias kann einem Child eine neue lo- gische Benennung zugeordnet werden
MERGE_MODE	Der Merge Mode gibt an ob ein Child Objekt mehrfach innerhalb eines Master Job-Laufes gestartet wird oder nicht. Es gibt folgende Optionen: No Merge, Failure, Merge Local und Merge Global

User Commands list job definition hierarchy

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
EST_NAME	Hierbei handelt es sich um die Exit State Translation
IGNORED_DEPENDENCIES	Hier kann eine Liste von Abhängigkeiten hinzugefügt werden, welche innerhalb dieser Parent-Child-Beziehung vom Child ignoriert werden soll
HIERARCHY_PATH	Der Path beschreibt die darüberliegende Ord- nerhierarchie eines Objektes. Alle übergeord- neten Ordner werden punktgetrennt angezeigt
STATES	Der State ist der aktuelle Status des Jobs
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.13.: Beschreibung der Output-Struktur des list job definition hierarchy Statements

list named resource

User Commands

list named resource

Zweck

Das list named resource Statement wird eingesetzt um eine Liste von allen definierte **Zweck** Named Resources zu bekommen.

Syntax

Die Syntax des list named resource Statements ist

Syntax

```
list named resource [ resourcepath ] [ with WITHITEM {, WITHITEM } ]
WITHITEM:
     expand = none
    expand = < ( id {, id} ) | all >
    FILTERTERM {or FILTERTERM}
FILTERTERM:
FILTERITEM {and FILTERITEM}
FILTERITEM:
     ( FILTERTERM { or FILTERTERM} )
     name like string
     not ( FILTERTERM {or FILTERTERM} )
     usage in ( RESOURCE_USAGE {, RESOURCE_USAGE} )
RESOURCE_USAGE:
     category
     static
     synchronizing
     system
```

Beschreibung

Mit dem list named resource Statement bekommt man eine Liste aller definierten Named Resources. Wenn eine Resource spezifiziert wird, wir diese Named Resource sowie, wenn es sich bei der Named Resource um eine Category handelt, alle Children. Die Liste der Named Resources kann durch die Spezifikation eines Filters entsprechend verkürzt werden.

Beschreibung

list named resource

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

filter Durch die Spezifikation von Filters können Named Resources nach Name und oder Usage gefiltert werden. Für die Syntax von Regular Expressions wird auf die ofiizielle Java Dokumentation verwiesen.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
USAGE	Die Usage gibt an um welchen Typ Resource es sich handelt
RESOURCE_STATE_PROFILE	Es handelt sich hier um das zur Resource zuge- ordnete Resource State Profile
FACTOR	Dies ist der standard Faktor mit der Resource Requirement Amounts multipliziert wird, wenn bei der betreffende Resource nichts anders spez- ifiziert ist
SUBCATEGORIES	Dies ist die Anzahl Categories die als Children unterhalb der gezeigten Named Resource vorhanden sind
RESOURCES	Das sind die Instanzen der Named Resource
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.14.: Beschreibung der Output-Struktur des list named resource Statements

list resource state definition

Zweck

Der Zweck der list resource state definition ist es eine Liste von allen definierten Zweck Resource States zu bekommen.

Syntax

Die Syntax des list resource state definition Statements ist

Syntax

list resource state definition

Beschreibung

Mit dem list resource state definition Statement bekommt man eine Liste aller Beschreibung definierten Resource States.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.15.: Beschreibung der Output-Struktur des list resource state definition Statements

list resource state mapping

list resource state mapping

Zweck

Zweck

Das list resource state mapping Statement wird eingesetzt um ein Liste von allen definierten Resource State Mappings zu bekommen.

Syntax

Syntax

Die Syntax des list resource state mapping Statements ist

list resource state mapping

Beschreibung

Beschreibung

Mit dem list resource state mapping Statement bekommt man eine Liste aller definierten Resource State Mappings.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
	adi dieses Objekt chinar

Tabelle 14.16.: Beschreibung der Output-Struktur des list resource state mapping Statements

list resource state profile

Zweck

Das list resource state profile Statement wird eingesetzt um eine Liste von allen Zweck derzeit definierten Resource State Profiles zu bekommen.

Syntax

Die Syntax des list resource state profile Statements ist

Syntax

list resource state profile

Beschreibung

Mit dem list resource state profile Statement bekommt man eine Liste aller definierte Beschreibung Resource State Profiles.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
INITIAL_STATE	Dieses Feld definiert den initialen Status der Re-
	source. Dieser Resource State muss nicht in der
	Liste gültiger Resource States vorhanden sein
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 14.17.: Beschreibung der Output-Struktur des list resource state profile Statements

list schedule

list schedule

Zweck

Zweck

Das list schedule Statement wird eingesetzt um eine Liste von allen definierten Zeitplänen zu erhalten.

Syntax

Syntax

Die Syntax des list schedule Statements ist

list schedule schedulepath [with EXPAND]

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Beschreibung

Das list schedule Statement liefert eine Liste mit dem angegebenen Schedule sowie aller seiner Children.

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
INTERVAL	Der Name des zum Schedule gehörende Inter-
	val
Fortsetzung auf der nächsten	Seite

list schedule User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TIME_ZONE	Die Zeitzone in der der Schedule gerechnet werden soll
ACTIVE	Dieses Feld gibt an ob der Schedule als active markiert ist
EFF_ACTIVE	Dieses Feld gibt an ob der Schedule tatsächlich active ist. Dies kann aufgrund der hierarchische Anordnug von "active" abweichen
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.18.: Beschreibung der Output-Struktur des list schedule Statements

list scheduled event

list scheduled event

Zweck

Zweck

Der Zweck des list scheduled event Statements ist, eine Liste von allen definierten Scheduled Events zu bekommen.

Syntax

Syntax

Die Syntax des list scheduled event Statements ist

list scheduled event

Beschreibung

Beschreibung

Mit dem list scheduled event Statement bekommt man eine Liste aller definierten Scheduled Events.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCHEDULE	Der Schedule der den Zeitplan für den Scheduled Event bestimmt
EVENT	Der Event der ausgelöst wird
ACTIVE	Dieses Flag gibt an ob der Scheduled Event als active gekennzeichnet ist
EFF_ACTIVE	Dieses Flag gibt an ob der Scheduled Event auch tatsächlich active ist
BROKEN	Mittels des Broken Feldes kann geprüft werden, ob beim Submit des Jobs ein Fehler aufgetreten ist
ERROR_CODE	Im Feld Error Code wird, falls ein Fehler bei der Ausführung des Jobs im Time Schedul- ing aufgetreten ist, der übermittelte Fehlercode angezeigt. Ist kein Fehler aufgetreten, bleibt das Feld leer

Toriscizarig dar der nachsten sene

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ERROR_MSG	Im Feld Error Message wird, falls ein Fehler bei der Ausführung des Jobs im Time Scheduling aufgetreten ist, die übermittelte Fehlermeldung angezeigt. Ist kein Fehler aufgetreten, bleibt das Feld leer
LAST_START	Hier wird der letzte Ausführungszeitpunkt des Jobs durch das Scheduling System angezeigt
NEXT_START	Hier wird der nächste geplante Ausführungszeitpunkt des Tasks durch das Scheduling System angezeigt
NEXT_CALC	Der nächste Zeitpunkt zu dem eine Neuberechnung erfolgen soll
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
BACKLOG_HANDLING	Das Backlog Handling beschreibt der Umgang mit Events die zu Downtimes ausgelöst hätten werden sollen
SUSPEND_LIMIT	Das Suspend Limit gibt an nach welcher Verspätung ein Job als suspended submitted wird
EFFECTIVE_SUSPEND_LIMIT	Das Suspend Limit gibt an nach welcher Verspätung ein Job als suspended submitted wird
CALENDAR	Dieses Flag gibt an, ob Kalendereinträge erzeugt werden
CALENDAR_HORIZON	Die definierte Länge der Periode in Tagen für die ein Kalender erstellt wird
EFFECTIVE_CALENDAR_HOR	т⊅о́м effektive Länge der Periode in Tagen für die ein Kalender erstellt wird

Tabelle 14.19.: Beschreibung der Output-Struktur des list scheduled event Statements

list scope

list scope

Zweck

Zweck

Das list scope Statement wird eingesetzt um eine Liste aller definierten Scopes zu bekommen.

Syntax

Syntax

Die Syntax des list scope Statements ist

```
list < scope serverpath | job server serverpath > [ with EXPAND ]
```

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Beschreibung

Mit dem list scope Statement bekommt man die Liste des beantragten Scopes mit seinen Children angezeigt.

expand Mit der **expand** Option kann die Hierarchie nach unten sichtbar gemacht werden. Dazu werden die Ids von den Knoten deren Children sichtbar sein sollen in der Liste spezifiziert. Falls **none** als expand Option spezifiziert wird, wird nur die Ebene unterhalb des beantragten Knoten sichtbar gemacht.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
TYPE	Der Typ des Scopes
IS_TERMINATE	Dieser Flag zeigt an ob ein Terminierungsauf-
	trag vorliegt
Fortsetzung auf der nächs	ten Seite

list scope

User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
HAS_ALTERED_CONFIG	Die Konfiguration im Server weicht von der aktuellen im Jobserver ab
IS_SUSPENDED	Zeigt an ob der Scope suspended ist
IS_ENABLED	Nur wenn der Enable Flag auf Yes gesetzt ist kann sich der Jobserver am Server anmelden
IS_REGISTERED	Gibt an ob der Jobserver ein Register Befehl gesendet hat
IS_CONNECTED	Zeigt an ob der Jobserver Connected ist
STATE	Hierbei handelt es sich um den aktuellen Status der Resource in diesem Scope
PID	Bei dem PID handelt es sich um die Prozess Identifikationsnummer des Jobserverprosesses auf dem jeweiligen Hostsystem
NODE	Der Node gibt an auf welchem Rechner der Jobserver läuft. Dieses Feld hat einen rein dokumentativen Charakter
IDLE	Die Zeit die seit dem letzten Befehl vergangen ist. Das gilt nur für Jobserver
NOPDELAY	Die Zeit die ein Jobserver wartet nach einem "NOP"
ERRMSG	Hierbei handelt es sich um die zuletzt ausgegebene Fehlermeldung
SUBSCOPES	Die Anzahl Scopes und Jobserver die unter diesem Scope vorhanden sind
RESOURCES	Hier werden die Resourcen die in diesem Scope vorhanden sind angezeigt
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 14.20.: Beschreibung der Output-Struktur des list scope Statements

list session

list session

Zweck

Zweck

Das list session Statement wird eingesetzt um eine Liste von den connected sessions zu bekommen.

Syntax

Syntax

Die Syntax des list session Statements ist

list session

Beschreibung

Beschreibung

Mit dem list session Statement bekommt man eine Liste der connected Sessions.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
THIS	Die aktuelle Session wird in diesem Feld mit einem Asterisk (*) gekennzeichnet
SESSIONID	Die serverinterne Id der Session
PORT	Der TCP/IP Portnummer an dem die Session connected ist
START	Zeitpunkt an dem die Connection hergestellt wurde
TYPE	Type der Connection: User, Jobserver oder Job
USER	Name des connecting User, Jobserver oder Job (Jobid)
UID	Id des Users, Jobservers oder Jobs
IP	IP-Adresse der connecting Sessions
TXID	Nummer der letzte Transaktion die von der Ses-
	sion ausgeführt wurde
IDLE	Die Anzahl Sekunden seit dem letzten Statement einer Session
Fortsetzung auf der nächsten S	Seite

232 I list session

list session User Commands

Fortsetzung der vorherig	gen Seite
Feld	Beschreibung
STATE	Der Status der Session. Dies ist einer der folgenden: IDLE (keine Aktivität), QUEUED (Statement wartet auf Ausführung), ACTIVE (Statement wird gerade ausgeführt), COMMITTING (Änderungen einer schreibenden Transaktion werden geschrieben), CONNECT-
TIMEOUT	ED (noch nicht authentifiziert) Die Idle Zeit nach der die Session automatisch
INFORMATION	disconnected wird Zusätzliche Information zu der Session (option-
STATEMENT	al) Das Statement, dass gerade ausgeführt wird
WAIT	Dieses Feld wurde noch nicht beschrieben

Tabelle 14.21.: Beschreibung der Output-Struktur des list session Statements

list trigger

list trigger

Zweck

Zweck

Das list trigger Statement wird eingesetzt um eine Liste von definierten Trigger zu bekommen.

Syntax

Syntax

Die Syntax des list trigger Statements ist

list trigger

list trigger for folderpath

list trigger of folderpath

list trigger for CT_OBJECT

CT_OBJECT:

job definition folderpath

Beschreibung

Beschreibung

Mit dem list trigger Statement bekommt man eine Liste aller definierten Trigger.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Feld	Beschreibung				
ID	Die Nummer des Repository Objektes				
NAME	Der Name des Objektes				
OBJECT_TYPE	Der Typ des Objektes in dem der Trigger definiert ist				
Fortsetzung auf der nächsten	Seite				

list trigger

User Commands

Fortsetzung der vorherigen Seite				
Feld	Beschreibung			
OBJECT_SUBTYPE	Der Subtyp des Objektes in dem der Trigger definiert ist			
OBJECT_NAME	Kompletter Pfadname des Objektes in dem der Trigger definiert ist			
ACTIVE	Der Flag gibt an ob der Trigger momentan aktiv ist			
ACTION	Typ der ausgelöste Aktion: SUBMIT oder RE- RUN			
STATES	Eine Liste mit States die zum auslösen des Trig- gers führen			
SUBMIT_TYPE	Der Objekttyp der submitted wird, wenn getriggert wird			
SUBMIT_NAME	Name der Job Definition die submitted wird			
SUBMIT_SE_OWNER	Der Besitzer des Objektes der submitted wird			
SUBMIT_PRIVS	Die Privilegien auf das zu submittende Objekt			
MAIN_TYPE	Typ des main Jobs (Job/Batch)			
MAIN_NAME	Name des main Jobs			
MAIN_SE_OWNER	Owner des main Jobs			
MAIN_PRIVS	Privilegien auf den main Job			
PARENT_TYPE	Typ des parent Jobs (Job/Batch)			
PARENT_NAME	Name des parent Jobs			
PARENT_SE_OWNER	Owner des parent Jobs			
PARENT_PRIVS	Privilegien auf den parent Job			
TRIGGER_TYPE	Der Trigger Typ der beschreibt wann gefeuert wird			
MASTER	Zeigt an ob der Trigger einen Master oder ein Child submitted			
IS_INVERSE	Zeigt an ob der Trigger einen Master oder ein Child submitted			
SUBMIT_OWNER	Im Falle eines inverse Triggers gehört der Trigger dem getriggerten Job. Der Trigger kann so als Art Callback Funktion gesehen werden. Der Flag hat keinen Einfluß auf die Funktion des Triggers.			
Fortsetzung auf der nächste	n Seite			

list trigger

Fortsetzung der vorherigen Seite						
Feld	Beschreibung					
IS_CREATE	Zeigt an ob der Trigger auf create Events reagiert					
IS_CHANGE	Zeigt an ob der Trigger auf change Events reagiert					
IS_DELETE	Zeigt an ob der Trigger auf delete Events reagiert					
IS_GROUP	Zeigt an ob der Trigger die Events als Gruppe behandelt					
MAX_RETRY	Die maximale Anzahl von Trigger Auslösungen in einem einzelnen Submitted Entity					
SUSPEND	Spezifiziert ob das submittete Objekt suspended wird					
RESUME_AT	Zeitpunkt des automatischen Resume					
RESUME_IN	Anzahl Zeiteinheite bis zum automatischen Resume					
RESUME_BASE	Zeiteinheit zu resume_in					
WARN	Spezifiziert ob eine Warnung ausgegeben werden muss wenn das Feuerlimit erreicht ist					
LIMIT_STATE	Spezifiziert den Status der vom auslösenden Jobs angenommen wird, wenn das Fire Limit erreicht wird. Hat der Job bereit einen finalen Status, wird diese Einstellung ignoriert. Steht der Wert auf NONE, wird keine Statusänderung vorgenommen.					
CONDITION	Konditionaler Ausdruck um die Trigger Kondition zu definieren					
CHECK_AMOUNT	Die Menge der CHECK_Base Einheiten um die Kondition bei nicht Synchronen Trigger zu überprüfen					
CHECK_BASE	Einheiten für den CHECK_AMOUNT					
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält					
TAG	Dieses Feld wurde noch nicht beschrieben					
COMMENT	Kommentar zum Objekt, wenn vorhanden					
COMMENTTYPE	Typ des Kommentars					

Tabelle 14.22.: Beschreibung der Output-Struktur des list trigger Statements

list user User Commands

list user

Zweck

Das list user Statement wird eingesetzt um eine Liste von allen definierten Be- Zweck nutzer zu erhalten.

Syntax

Die Syntax des list user Statements ist

Syntax

list user

Beschreibung

Mit dem list user Statement bekommt man eine Liste aller definierten Benutzer.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung			
ID	Die Nummer des Repository Objektes			
NAME	Der Name des Objektes			
IS_ENABLED	Flag das anzeigt ob es dem Benutzer erlaubt ist sich anzumelden			
DEFAULT_GROUP	Die Default Gruppe der Benutzer die die Eigentümer des Objektes benutzen			
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält			

Tabelle 14.23.: Beschreibung der Output-Struktur des list user Statements

list user | 237

15. move commands

move folder

move folder

Zweck

Zweck

Das move folder Statement wird eingesetzt um den Folder umzubenennen und/oder ihn an einer anderen Stelle in der Ordnerhierarchie zu verschieben.

Syntax

Syntax

Die Syntax des move folder Statements ist

move folder folderpath to folderpath

Beschreibung

Beschreibung

Der move folder Befehl verschiebt den angegebenen Folder an einer anderen Stelle oder er benennt ihn um.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

move job definition

User Commands

move job definition

Zweck

Das move job definition Statement wird eingesetzt um ein Scheduling Entity Zweck Objekt umzubenennen und/oder ihn in einem anderen Folder zu verschieben.

Syntax

Die Syntax des move job definition Statements ist

Syntax

move job definition folderpath to folderpath

Beschreibung

Der move job definition Befehl verschiebt die angegebene Job Definition in den angegebenen Folder. Wenn der Zielfolder nicht existiert, wird das letzte Glied des vollqualifizierten Namen als neuer Name für die Job Definition interpretiert. Die Beziehungen zu anderen Objekten werden nicht geändert.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

move named resource

Zweck

Zweck

Das move named resource Statement wird eingesetzt um die Named Resource umzubenennen und/oder um die Resource in eine andere Kategorie zu verschieben.

Syntax

Syntax

Die Syntax des move named resource Statements ist

move named resource resourcepath to resourcepath

Beschreibung

Beschreibung

Das move named resource Statement wird benutzt um eine Named Resource umzubenennen oder Kategorien neu zu ordnen.

Wenn eine Named Resource verschoben wird, muss das spezifizierte Ziel eine Kategorie sein oder es darf nicht existieren und sein Parent muss eine Kategorie sein.

Ausgabe

Ausgabe

move schedule

User Commands

move schedule

Zweck

Das move schedule Statement wird eingesetzt um den Zeitplan umzubenennen Zweck oder ihn zu einem anderen Ort in der Hierarchie zu verschieben.

Syntax

Die Syntax des move schedule Statements ist

Syntax

Beschreibung

Ausgabe

move schedule schedulepath . schedulename to schedulepath

Beschreibung

Der move Schedule Befehl verschiebt den angegebenen Schedule an einem anderen Ort und/oder er benennt ihn um.

Ausgabe

move scope

move scope

Zweck

Zweck

Das move scope Statement wird eingesetzt um einen Scope umzubenennen und/oder an anderer Stelle in der Scope-Hierarchie zu verschieben.

Syntax

Syntax

Die Syntax des move scope Statements ist

move < scope server path | job server server path > to server path

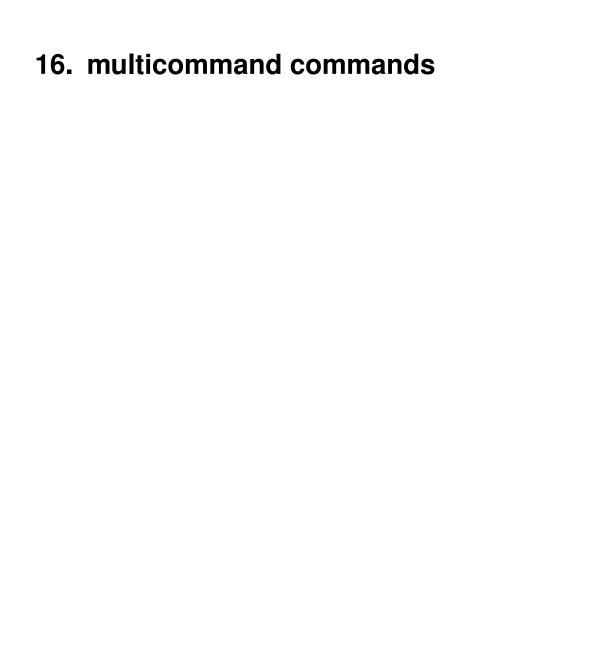
Beschreibung

Beschreibung

Der move scope Befehl verschiebt den angegebenen Scope an einem anderen Ort und/oder er benennt ihn um.

Ausgabe

Ausgabe



multicommand

multicommand

User Commands

Zweck

Zweck

Der Zweck des multicommands ist es mehrere SDMS-Kommandos als Einheit zuzuführen.

Syntax

Syntax

Die Syntax des multicommand Statements ist

begin multicommand commandlist end multicommand

begin multicommand commandlist end multicommand rollback

Beschreibung

Beschreibung

Mit den multicommands ist es möglich mehrere SDMS-Kommandos zusammen, d.h. in einer Transaktion aus zu führen. Damit wird gewährleistet, dass entweder alle Statements fehlerfrei ausgeführt werden, oder nichts passiert. Desweiteren wird die Transaktion nicht von anderen schreibenden Transaktionen unterbrochen. Wird das Keyword rollback spezifiziert, wird die Transaktion am Ende der Verarbeitung rückgängig gemacht. Auf dieser Weise kann getestet werden ob die Statements (technisch) korrekt verarbeitet werden können.

Ausgabe

Ausgabe

17. register commands

register

register

Zweck

Zweck

Das register Statement wird eingesetzt um den Server zu benachrichtigen, dass der Jobserver bereit ist Jobs auszuführen.

Syntax

Syntax

Die Syntax des register Statements ist

```
register serverpath . servername
with pid = pid [ suspend ]
```

register with pid = pid

Beschreibung

Beschreibung

Die erste Form wird vom Operator benutzt um das Aktivieren von Jobs auf dem spezifizierten Jobserver zu ermöglichen.

Die zweite Form wird vom Jobserver selbst benutzt um den Server über seine Bereitschaft Jobs auszuführen zu informieren.

Unabhängig davon ob der Jobserver connected ist oder nicht, werden Jobs für diesen Server eingeplant, es sei den der Jobserver ist suspended.

Siehe das Statement 'deregister' auf Seite 150.

pid Die pid Option liefert dem Server Informationen über die Prozess ID des Jobservers auf Betriebsebene.

suspend Die suspend Option bewirkt, dass der Jobserver in den Suspended Zustand überführt wird.

Ausgabe

Ausgabe

18. rename commands

rename environment

rename environment

Zweck

Zweck

Das rename environment Statement wird eingesetzt um den spezifizierten Environment umzubenennen.

Syntax

Syntax

Die Syntax des rename environment Statements ist

rename environment environmentname to environmentname

Beschreibung

Beschreibung

Das rename environment Statement wird benutzt um environments umzubenennen. Die Umbenennung eines Environments hat keinen Einfluss auf Funktionalitäten und dient nur der Übersichtlichkeit.

Ausgabe

Ausgabe

rename event

User Commands

rename event

Zweck

Der Zweck des rename event Statements ist es, dem spezifizierten Event einen Zweck anderen Namen zu geben.

Syntax

Die Syntax des rename event Statements ist

Syntax

Ausgabe

rename event eventname to eventname

Beschreibung

Mit dem rename event Statement vergibt man einem spezifizierten Event einen Beschreibung anderen Namen.

Ausgabe

rename exit state definition

rename exit state definition

Zweck

Zweck

Das rename exit state definition Statement wird eingesetzt um die spezifizierte Exit State Definition umzubenennen.

Syntax

Syntax

Die Syntax des rename exit state definition Statements ist

rename exit state definition statename to statename

Beschreibung

Beschreibung

Das rename exit state definition Statement wird benutzt um Exit State Definitions umzubenennen. Das Umbenennen einer Exit State Definition hat keinen Einfluss auf Funktionalitäten und dient nur der Übersichtlichkeit.

Ausgabe

Ausgabe

rename exit state mapping

Zweck

Das rename exit state mapping Statement wird eingesetzt um das spezifizierte Zweck Mapping umzubenennen.

Syntax

Die Syntax des rename exit state mapping Statements ist

Syntax

rename exit state mapping mappingname to profilename

Beschreibung

Das rename exit state mapping Statement wird benutzt um Exit State Mappings umzubenennen. Das Umbenennen eines Exit State Mappings hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename exit state profile

rename exit state profile

Zweck

Zweck

Das rename exit state profile Statement wird eingesetzt um den spezifizierten Profile umzubenennen.

Syntax

Syntax

Die Syntax des rename exit state profile Statements ist

rename exit state profile profilename to profilename

Beschreibung

Beschreibung

Das rename exit state profile Statement wird benutzt um Exit State Profiles umzubenennen. Das Umbenennen der Exit State Profiles hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Ausgabe

Ausgabe

rename folder

User Commands

rename folder

Zweck

Das Rename Folder Statement dient zum Umbenennen eines Folders.

Zweck

Syntax

Die Syntax des rename folder Statements ist

Syntax

rename folder folderpath **to** foldername

Beschreibung

Der rename folder Befehl benennt den angegebenen Folder um. Dies geschieht Beschreibung innerhalb des gleichen Parent Folders. Wenn bereits ein Objekt mit dem neuen Namen vorhanden ist, wird eine Fehlermeldung ausgegeben.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename footprint

rename footprint

Zweck

Zweck

Das rename footprint Statement wird eingesetzt um den spezifizierten Footprint umzubenennen.

Syntax

Syntax

Die Syntax des rename footprint Statements ist

rename footprint footprintname to footprintname

Beschreibung

Beschreibung

Mit dem rename footprint Statement vergibt man einem spezifizierten Footprint einen anderen Namen.

Ausgabe

Ausgabe

rename group

User Commands

rename group

Zweck

Das rename group Statement wird eingesetzt um den Namen einer Gruppe zu Zweck ändern, ohne das andere Eigenschaften davon betroffen werden .

Syntax

Die Syntax des rename group Statements ist

Syntax

rename group groupname to groupname

Beschreibung

Das rename group Statement wird benutzt um Gruppen umzubenennen. Das Umbenennen einer Gruppe hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename interval

rename interval

Zweck

Zweck

Das rename interval Statement wird eingesetzt um den spezifizierten Interval umzubenennen.

Syntax

Syntax

Die Syntax des rename interval Statements ist

rename interval intervalname to intervalname

Beschreibung

Beschreibung

Mit dem rename interval Statement vergibt man dem spezifizierten Interval einen anderen Namen.

Ausgabe

Ausgabe

rename job definition

User Commands

rename job definition

Zweck

Das Rename Job Definition Statement dient zum Umbenennen einer Job Defini- Zweck tion.

Syntax

Die Syntax des rename job definition Statements ist

Syntax

rename job definition folderpath to jobname

Beschreibung

Der rename job definition Befehl benennt die angegebene Job Definition um.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename named resource

rename named resource

Zweck

Zweck

Das Rename Named Resource Statement dient zum Umbenennen einer Named Resource.

Syntax

Syntax

Die Syntax des rename named resource Statements ist

rename named resource resourcepath to resourcename

Beschreibung

Beschreibung

Das move named resource Statement wird benutzt um eine Named Resource umzubenennen.

Ausgabe

Ausgabe

rename resource state definition

Zweck

Das rename resource state definition Statement wird eingesetzt um den Resource Zweck State umzubenennen.

Syntax

Die Syntax des rename resource state definition Statements ist

Syntax

rename resource state definition statename to statename

Beschreibung

Das rename resource state definition Statement wird benutzt um Resource State Definitions umzubenennen. Das Umbenennen einer Resource State Definition hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename resource state mapping

rename resource state mapping

Zweck

Zweck

Das rename resource state mapping Statement wird eingesetzt um dem spezifizierten Mapping einen neuen Namen zu geben.

Syntax

Syntax

Die Syntax des rename resource state mapping Statements ist

rename resource state mapping mappingname to profilename

Beschreibung

Beschreibung

Das rename resource state mapping Statement wird benutzt um Resource State Mappings umzubenennen. Das Umbenennen eines Resource State Mappings hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Ausgabe

Ausgabe

rename resource state profile

Zweck

Der Zweck des rename resource state profiles ist es, dem spezifizierten Resource Zweck State Profile einen neuen Namen zu geben.

Syntax

Die Syntax des rename resource state profile Statements ist

Syntax

rename resource state profile profilename to profilename

Beschreibung

Das rename resource state profile Statement wird benutzt um Resource State Profiles umzubenennen. Das Umbenennen eines Resource State Profiles hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename schedule

rename schedule

Zweck

Zweck Das Rename Schedule Statement dient zum Umbenennen eines Schedules.

Syntax

Syntax Die Syntax des rename schedule Statements ist

 $\textbf{rename schedule} \ schedule path. \ schedule name \ \textbf{to} \ schedule name$

Beschreibung

Beschreibung Der rename Schedule Befehl benennt den angegebenen Schedule um.

Ausgabe

rename scope

User Commands

rename scope

Zweck

Das Rename Scope Statement dient zum Umbenennen eines Scopes.

Zweck

Syntax

Die Syntax des rename scope Statements ist

Syntax

rename < **scope** *serverpath* | **job server** *serverpath* > **to** *scopename*

Beschreibung

Der rename scope Befehl benennt den angegebenen Scope um.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

rename trigger

rename trigger

Zweck

Zweck

Das rename trigger Statement wird eingesetzt um dem spezifizierten Trigger einen anderen Namen zu geben.

Syntax

Syntax

Die Syntax des rename trigger Statements ist

```
rename trigger triggername on <code>TRIGGEROBJECT</code> [ < noinverse | inverse > ] to triggername
```

TRIGGEROBJECT:

resource resourcepath in folderpath job definition folderpath object monitor objecttypename resource resourcepath in serverpath

Beschreibung

Beschreibung

Das rename trigger Statement wird benutzt um den Trigger umzubenennen. Das Umbenennen eines Triggers hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Ausgabe

Ausgabe

rename user

User Commands

rename user

Zweck

Das rename user Statement wird eingesetzt um den Benutzernamen umzube- Zweck nennen, ohne eine seiner anderen Eigenschaften zu ändern.

Syntax

Die Syntax des rename user Statements ist

Syntax

rename user username to username

Beschreibung

Das rename user Statement wird benutzt um Users umzubenennen. Das Umbenennen eines Users hat keinen Einfluss auf die Funktionalitäten und dient nur der Übersichtlichkeit.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

I 267

19. resume commands

resume

resume

Zweck

Zweck

Das resume Statement wird eingesetzt um den Jobserver zu reaktivieren. Siehe das suspend Statement auf Seite 372.

Syntax

Syntax

Die Syntax des resume Statements ist

resume serverpath

Beschreibung

Beschreibung

Mit dem resume Statement reaktiviert man einen Jobserver.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

270 l resume

20. select commands

select

select

Zweck

Zweck

Das select Statement wird eingesetzt um es dem Benutzer zu ermöglichen nahezu beliebige queries auszuführen.

Syntax

Syntax

Die Syntax des select Statements ist

select-statement [with WITHITEM {, WITHITEM}]

WITHITEM:

identifier category
identifier folder
identifier job
identifier scope
sort (signed_integer {, signed_integer})

Beschreibung

Beschreibung

Das Select Statement ermöglicht es nahezu beliebige Datenbank Select Statements vom Scheduling Server ausführen zu lassen. Für den Syntax des Select Statements wird dazu auf die Dokumentation des eingesetzten Datenbanksystem verwiesen.

Da das Absetzen von beliebigen Select Statements prinzipiell eine Sicherheitslücke darstellt, werden für dieses Statement Administratorrechte benötigt. Das heisst, dass nur Benutzer der Gruppe **admin** dieses Statement nutzen können.

Die Benutzung der Withitems führt zum Übersetzen von Ids nach Namen. Dies wird für alle hierarchisch strukturierte Objekttypen angeboten, da diese Operation mit SQL Mitteln nicht immer einfach durch zu führen ist. Es ist ebenfalls möglich die Ergebnismenge nach dem Ersetzen der Ids zu sortieren. Die Spalten nach den sortiert werden soll, werden durch ihre Position in der Ergebnismenge addressiert (zero based, d.h. die erste Spalte hat Nummer 0).

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

272 | select

21. set commands

set parameter

set parameter

Zweck

Zweck

Das set parameter Statement wird eingesetzt um den Wert des spezifizierten Parameters innerhalb des Kontext eines Jobs zu setzen.

Syntax

Syntax

Die Syntax des set parameter Statements ist

```
set parameter parametername = string {, parametername = string}
```

```
set parameter < on | of > jobid parametername = string {, parametername = string}
```

```
set parameter < on | of > jobid parametername = string {, parametername = string} identified by string
```

Beschreibung

Beschreibung

Mittels des set parameter Statements können Jobs oder Benutzer Parameterwerte in dem Kontext des Jobs setzen.

Falls die **identified by** Option spezifiziert ist, wird der Parameter nur dann gesetzt, wenn das Paar *jobid* und *string* eine Anmeldung ermöglichen wurden.

Ausgabe

Ausgabe

22. show commands

show comment

show comment

Zweck

Zweck

Das show comment Statement wird eingesetzt um den Kommentar zu dem spezifizierten Objekt anzuzeigen.

Syntax

Syntax

Die Syntax des show comment Statements ist

show comment on OBJECTURL

```
OBJECTURL:
```

distribution distributionname for pool resourcepath in serverpath environment environmentname exit state definition statename exit state mapping mappingname exit state profile profilename event eventname resource resourcepath in folderpath **folder** *folderpath* footprint footprintname group groupname interval intervalname job definition folderpath job jobid named resource resourcepath parameter parametername of PARAM_LOC resource state definition statename resource state mapping mappingname resource state profile profilename scheduled event schedulepath. eventname schedule schedulepath resource resourcepath in serverpath < **scope** *serverpath* | **job server** *serverpath* > trigger triggername on TRIGGEROBJECT [< noinverse | inverse >]

PARAM_LOC:

folder folderpath
| job definition folderpath

user username

< scope serverpath | job server serverpath >

TRIGGEROBJECT:

resource resourcepath in folderpath job definition folderpath object monitor objecttypename

resource resourcepath in serverpath

Beschreibung

Das show comment Statement dient der Anzeige des, zum spezifizierten Objekt, abgelegten Kommentars. Wenn kein Kommentar zu dem Objekt vorhanden ist, wird dies nicht als Fehler gesehen, sondern es wird eine leere Outputstruktur erzeugt und zurückgegeben. Diese leere Outputstruktur entspricht natürlich der unten beschriebenen Outputstruktur, so dass sie auch leicht von Programmen, ohne Ausnahmebehandlung, ausgewertet werden kann.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Systemweit eindeutige Objektnummer
TAG	Der Comment Tag ist eine Kopfzeile für das ak-
	tuelle Kommentarblock. Das Feld ist optional.
COMMENT	Der Kommentar zu dem spezifizierten Objekt
COMMENTTYPE	Typ des Kommentars, Text oder URL
CREATOR	Name des Benutzers der diesen Pool angelegt
	hat
CREATE_TIME	Zeitpunkt des Anlegens
CHANGER	Name des Benutzers der diesen Pool zuletzt
	geändert hat
CHANGE_TIME	Zeitpunkt der letzten Änderung
PRIVS	Abkürzung für die Privilegien die der anfra-
	gende Benutzer auf diesem Objekt hat

Tabelle 22.1.: Beschreibung der Output-Struktur des show comment Statements

show environment

show environment

Zweck

Zweck

Das show environment Statement wird eingesetzt um detaillierte Informationen über den spezifizierten Environment zu bekommen.

Syntax

Syntax

Die Syntax des show environment Statements ist

show environment *environmentname* [**with** EXPAND]

```
EXPAND:

expand = none

expand = < ( id {, id} ) | all >
```

Beschreibung

Beschreibung

Mit dem show environment Statement bekommt man ausführliche Informationen über das spezifizierte Environment.

expand Da die Anzahl Job Definitions in der Tabelle JOB_DEFINITIONS sehr groß werden kann, werden sie per Default nicht angezeigt. Wenn die Option **expand = all** benutzt wird, werden alle Job Definitions sowie die Folder in die sie liegen samt Folderhierarchie ausgegeben. Durch die Spezifikation einzelner (Folder) Ids, können einzelne Pfade der Hierarchie selektiert werden.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Environments
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
Fortsetzung auf der näch	sten Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
RESOURCES	Tabelle von statischen Resourcen die dieses Environment formen
	Siehe auch Tabelle 22.3 auf Seite 279
JOB_DEFINITIONS	Tabelle von Jobs und Folder die dieses Environ- ment nutzen
	Siehe auch Tabelle 22.4 auf Seite 280

Tabelle 22.2.: Beschreibung der Output-Struktur des show environment Statements

RESOURCES Das Layout der RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NR_NAME	Kompletter Pfadname von statischen Named
	Resourcen
CONDITION	Die Condition die zur Belegung erfüllt sein
	muss
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 22.3.: Output-Struktur der show environment Subtabelle

JOB_DEFINITIONS Das Layout der JOB_DEFINITIONS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
SE_PATH	Kompletter Folder Pfadname von Job Definitio-
	nen oder Folder
Fortsetzung auf der näch	hsten Seite

User Commands show environment

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TYPE	Der Objekttyp. Die möglichen Werte sind FOLDER und JOB_DEFINITION
ENV	Ein Asterisk zeigt an, dass das aktuelle Environ- ment hier spezifiziert wurde
HAS_CHILDREN	true bedeutet, dass es weiter unten im Baum noch Environment Benutzer gibt
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.4.: Output-Struktur der show environment Subtabelle

show event

User Commands

show event

Zweck

Das show event Statement wird eingesetzt um detaillierte Informationen über Zweck das spezifizierte Event zu bekommen.

Syntax

Die Syntax des show event Statements ist

Syntax

show event eventname

Beschreibung

Mit dem show event Statement bekommt man ausführliche Informationen über Beschreibung das spezifizierte Event.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Show Events
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCHEDULING_ENTITY	Batch oder Job der submitted wird wenn dieses Event eintritt
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PARAMETERS	Parameter die beim submit des Jobs oder Batches benutzt werden
	Siehe auch Tabelle 22.6 auf Seite 282
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

show event

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.5.: Beschreibung der Output-Struktur des show event Statements

PARAMETERS Das Layout der PARAMETERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
KEY	Name des Parameters
VALUE	Wert des Parameters

Tabelle 22.6.: Output-Struktur der show event Subtabelle

show exit state definition

Zweck

Das show exit state definition Statement wird eingesetzt um detaillierte Informa- *Zweck* tionen über die spezifizierte Exit State Definition zu bekommen.

Syntax

Die Syntax des show exit state definition Statements ist

Syntax

show exit state definition statename

Beschreibung

Mit dem show exit state definition Statement bekommt man ausführliche Informationen über die spezifizierte Exit State Definition.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name der Exit State Definiton
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt
	geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 22.7.: Beschreibung der Output-Struktur des show exit state definition Statements

show exit state mapping

show exit state mapping

Zweck

Zweck

Das show exist state mapping Statement wird eingesetzt um detaillierte Informationen über das spezifizierte Mapping zu bekommen.

Syntax

Syntax

Die Syntax des show exit state mapping Statements ist

show exit state mapping mappingname

Beschreibung

Beschreibung

Mit dem show exit state mapping Statement bekommt man ausführliche Informationen über das spezifizierte Mapping.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Exit State Mappings
COMMENT	Ein Kommentar, von Benutzer frei wählbar
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
RANGES	Die Zuordnung der jeweiligen Wertebereiche, in einer Tabelle dargestellt

show exit state mapping

User Commands

Fortsetzung der vo	orherigen Seite
Feld	Beschreibung
	Siehe auch Tabelle 22.9 auf Seite 285

Tabelle 22.8.: Beschreibung der Output-Struktur des show exit state mapping Statements

RANGES Das Layout der RANGES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ECR_START	Untere Grenze des Bereiches (inklusiv)
ECR_END	Obere Grenze des Bereiches (inklusiv)
ESD_NAME	Name des Exit Status auf dem dieser Bereich abgebildet wird

Tabelle 22.9.: Output-Struktur der show exit state mapping Subtabelle

show exit state profile

Zweck

Zweck

Das show exist state profile Statement wird eingesetzt um detaillierte Informationen über den spezifizierten Profile zu bekommen.

Syntax

Syntax

Die Syntax des show exit state profile Statements ist

show exit state profile profilename

Beschreibung

Beschreibung

Mit dem show exit state profile Statement bekommt man ausführliche Informationen über den spezifizierten Profile.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
DEFAULT_ESM_NAME	Default Exit State Mapping ist aktiv wenn der Job selbst nichts anderes angibt
IS_VALID	Flag Anzeige über die Gültigkeit dieses Exit State Profiles
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

show exit state profile

User Commands

Fortsetzung der vor	herigen Seite
Feld	Beschreibung
STATES	Tabelle beinhaltet Exit States die für dieses Pro- file gültig sind
	Siehe auch Tabelle 22.11 auf Seite 287

Tabelle 22.10.: Beschreibung der Output-Struktur des show exit state profile Statements

STATES Das Layout der STATES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
PREFERENCE	Die Präferenz die Verbindung der Child Exit States zu kontrollieren
ТҮРЕ	Zeigt an ob der State FINAL, PENDING oder RESTARTABLE ist
ESD_NAME	Name der Exit State Definition
IS_UNREACHABLE	Zeigt an dass diese Exit State benutzt wird, wenn ein Job unreachable wird
IS_BROKEN	Zeigt an dass diese Exit State benutzt wird, wenn ein Job fehlerhaft ist
IS_BATCH_DEFAULT	Zeigt an dass diese Exit State benutzt wird, wenn ein Batch oder Milestane keine Kinder hat
IS_DEPENDENCY_DEFAULT	Zeigt an dass diese Exit State benutzt wird, wenn bei der Dependency Definition die State Selection DEFAULT gewält wurde.

Tabelle 22.11.: Output-Struktur der show exit state profile Subtabelle

show folder

show folder

Zweck

Zweck

Das show folder Statement wird eingesetzt um detaillierte Informationen über den spezifizierten Folder zu bekommen.

Syntax

Syntax

Die Syntax des show folder Statements ist

show folder folderpath

Beschreibung

Beschreibung

Mit dem show folder Statement bekommt man ausführliche Informationen über den spezifizierten Folder.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Folders
OWNER	Die Gruppe die Eigentümer des Objektes ist
TYPE	Der Objekttyp der immer im Folder enthalten ist
ENVIRONMENT	Der Name des optionalen Environments
INHERIT_PRIVS	Vom übergeordneten Ordner zu erbende Privilegien
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat

288 I show folder

show folder User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
PARAMETERS	Die Parameter Tabelle zeigt alle definierten Constanten für diesen Folder an Siehe auch Tabelle 22.13 auf Seite 290
DEFINED_RESOURCES	Die defined Resources Tabelle zeigt alle Resource Instanzen an, die für diesen Folder definiert sind Siehe auch Tabelle 22.14 auf Seite 292

Tabelle 22.12.: Beschreibung der Output-Struktur des show folder Statements

PARAMETERS Das Layout der PARAMETERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
EXPORT_NAME	Der Export Name definiert den Namen unter dem der Wert des Parameters in die Prozes- sumgebung exportiert wird.
TYPE	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
EXPRESSION	Expression für den Parameter des Typs EX- PRESSION
DEFAULT_VALUE	Der Default Value des Parameters
REFERENCE_TYPE	Typ des Objektes auf das referenziert wird
REFERENCE_PATH	Der Pfad des Objektes auf das referenziert wird
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes auf das referenziert wird
REFERENCE_PARAMETER	Name des Parameters auf den referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
ID	Die Nummer des Repository Objektes
Fortsetzung auf der nächsten	Seite

I 289

show folder

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
NAME	Name des Parameters
TYPE	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
REFERENCE_TYPE	Typ des Objektes welches den Parameter referenziert
REFERENCE_PATH	Der Pfad des Objektes welches den Parameter referenziert
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes welches den Parameter referenziert
REFERENCE_PARAMETER	Name des Parameters auf dem referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.13.: Output-Struktur der show folder Subtabelle

DEFINED_RESOURCES Das Layout der DEFINED_RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NR_ID	Id der Named Resource
NAME	Name der Named Resource
USAGE	Hierbei handelt es sich um den Gebrauch der Named Resource (STATIC, SYSTEM oder SYN- CHRONIZING)
NR_PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf diese Named Resource enthält
TAG	Der Tag ist ein optionaler Kurzname für die Resource
OWNER	Die Gruppe die Eigentümer des Objektes ist
LINK_ID	Id der Ressource auf die verwiesen wird
LINK_SCOPE	Scope der Ressource auf die verwiesen wird
STATE	Der Resource State in dem sich die Resource befindet

show folder User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
REQUESTABLE_AMOUNT	Die Menge der Resourcen die maximal von
	einem Job angefordert werden darf
AMOUNT	Die aktuelle Menge die zur Verfügung steht
FREE_AMOUNT	Die freie Menge die allokiert werden darf
TOTAL_FREE_AMOUNT	Free amount available for allocations including free amount of pooled resources if it is a pool
IS_ONLINE	Hierbei handelt es sich um den Verfüg- barkeitsstatus der Resource
FACTOR	Dies ist der Korrekturfaktor mit dem ange- forderte Amounts multipliziert werden
TIMESTAMP	Der Timestamp gibt die Zeit des letzten Statuswechsels einer Resource an
SCOPE	Der Scope in dem die Resource angelegt ist
MANAGER_ID	Id des Managing Pools
MANAGER_NAME	Name des Managing Pools
MANAGER_SCOPENAME	Name des Scopes in dem der managing Pool an-
	gelegt wurde
HAS_CHILDREN	Flag das anzeigt ob ein Pool Child Resources/Pools managed. Wenn es kein Pool ist, ist es immer FALSE
POOL_CHILD	Dieses Flag zeigt an ob die gezeigte Resource ein Child vom Pool ist
TRACE_INTERVAL	Das trace Interval ist die minimale Zeit zwischen dem Scheiben von Trace Records in Sekunden
TRACE_BASE	Die trace Base ist es die Basis für den Auswertungszeitraum (B)
TRACE_BASE_MULTIPLIER	Der base Multiplier bestimmt den Multiplika- tionsfaktor (M) von der trace Base
TD0_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^0$ Sekunden
TD1_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^1$ Sekunden
TD2_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^2$ Sekunden
LW_AVG	Die durchschnittliche Resourcebelegung seit dem letzten Schreiben eines Trace Records
Fortsetzung auf der nächsten S	eite

User Commands show folder

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
LAST_WRITE	Zeitpunkt des letzen Schreibens eines Trace Records
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.14.: Output-Struktur der show folder Subtabelle

show footprint

User Commands

show footprint

Zweck

Das show footprint Statement wird eingesetzt um detaillierte Informationen Zweck über den spezifizierten Footprint zu bekommen.

Syntax

Die Syntax des show footprint Statements ist

Syntax

show footprint *footprintname* [**with** EXPAND]

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Mit dem show footprint Statement bekommt man ausführliche Informationen Beschreibung über den spezifizierten Footprint.

expand Da die Anzahl Job Definitions in der Tabelle JOB_DEFINITIONS sehr groß werden kann, werden sie per Default nicht angezeigt. Wenn die Option **expand = all** benutzt wird, werden alle Job Definitions sowie die Folder in die sie liegen samt Folderhierarchie ausgegeben. Durch die Spezifikation einzelner (Folder) Ids, können einzelne Pfade der Hierarchie selektiert werden.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Footprints
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
Fortsetzung auf der näch	sten Seite

show footprint

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
RESOURCES	Tabelle von system Resourcen die diesen Foot- print formen
	Siehe auch Tabelle 22.16 auf Seite 294
JOB_DEFINITIONS	Tabelle von Job Definitionen die diesen Foot-
	print nutzen
	Siehe auch Tabelle 22.17 auf Seite 295

Tabelle 22.15.: Beschreibung der Output-Struktur des show footprint Statements

RESOURCES Das Layout der RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
RESOURCE_NAME	Voll qualifizierter Pfadname von System Named Resourcen
AMOUNT	Menge der Resourceeinheiten die allokiert werden
KEEP_MODE	Keep Mode spezifiziert wann die Resource freigegeben wird (FINISH, JOB_FINAL oder FINAL)

Tabelle 22.16.: Output-Struktur der show footprint Subtabelle

JOB_DEFINITIONS Das Layout der JOB_DEFINITIONS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
SE_PATH	Ordner Pfad Name des Objektes
Fortsetzung auf der nächsten Seite	

show footprint User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TYPE	Typ des Objektes
HAS_CHILDREN	true bedeutet, dass es weiter unten im Baum noch Environment Benutzer gibt
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.17.: Output-Struktur der show footprint Subtabelle

show group

show group

Zweck

Zweck

Das show group Statement wird eingesetzt um detailierte Informationen über die spezifizierte Gruppe zu bekommen.

Syntax

Syntax

Die Syntax des show group Statements ist

show group groupname

Beschreibung

Beschreibung

Mit dem show group Statement bekommt man ausführliche Informationen über die spezifizierten Gruppe.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name der Gruppe
COMMENTTYPE	Typ des Kommentars
COMMENT	Kommentar zum Objekt, wenn vorhanden
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
MANAGE_PRIVS	Tabelle der manage Privilegien
	Siehe auch Tabelle 22.19 auf Seite 297
USERS	Tabelle der Benutzergruppen
Fortsetzung auf der nächsten S	eite

show group User Commands

Fortsetzung der vo	orherigen Seite
Feld	Beschreibung
	Siehe auch Tabelle 22.20 auf Seite 297

Tabelle 22.18.: Beschreibung der Output-Struktur des show group Statements

MANAGE_PRIVS Das Layout der MANAGE_PRIVS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 22.19.: Output-Struktur der show group Subtabelle

USERS Das Layout der USERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
UID	Id des Users
NAME	Der Name des Objektes
IS_ENABLED	Dieses Flag teilt dem Benutzer mit ob er verbunden werden kann
DEFAULT_GROUP	Die Default Gruppe von diesem Benutzer
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.20.: Output-Struktur der show group Subtabelle

show interval

show interval

Zweck

Zweck

Das show interval Statement wird eingesetzt um detaillierte Informationen über den Intervall zu bekommen.

Syntax

Syntax

Die Syntax des show interval Statements ist

```
show interval intervalname
[ with expand [ <= datetime | = datetime - datetime > ] ]
```

Beschreibung

Beschreibung

Das show interval Statement zeigt detaillierte Information zu einem Intervall. Ohne expand Klausel werden keine steigende Fanken (Edges) angezeigt. Mit dem expand Klausel kann eine Periode, für die die Edges gezeigt werden sollen, spezifiziert werden.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
STARTTIME	Der Anfangszeitpunkt des Intervalles
ENDTIME	Der Endzeitpunkt des Intervalles
BASE	Die Base ist eine Periode in der es Zeitblöcke bestimmter Längen gibt
DURATION	Hierbei handelt es sich um Zeitblöcke bes- timmter Längen
SYNCTIME	Der Zeitpunkt ab dem der Intervall beginnt. Ist der Synctime nicht gegeben wird der Starttime genommen

298 I show interval

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
INVERSE	Mit Inverse Wird die Auswahl umgekehrt
EMBEDDED	Der Named des embedded Intervalls
SELECTION	Mit Selection werden einzelne Blöcke selektiert
	Siehe auch Tabelle 22.22 auf Seite 299
FILTER	Name(n) der Intervalle die den Output dieses Intervalls weiter filtern (Multiplikation)
	Siehe auch Tabelle 22.23 auf Seite 300
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.21.: Beschreibung der Output-Struktur des show interval Statements

SELECTION Das Layout der SELECTION Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
VALUE	Nummer des ausgewählten Edges
PERIOD_FROM	Anfang der Periode in der alle auftretende Edges als ausgewählt gelten
PERIOD_TO	Ende der Periode in der alle auftretende Edges als ausgewählt gelten

Tabelle 22.22.: Output-Struktur der show interval Subtabelle

FILTER Das Layout der FILTER Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
Fortsetzung auf der	r nächsten Seite

show interval

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CHILD	Name des filternden Intervalls

Tabelle 22.23.: Output-Struktur der show interval Subtabelle

Syntax

show job

User Commands

show job

Zweck

Das show job Statement wird eingesetzt um detaillierte Informationen über den Zweck spezifizierten Job zu bekommen.

Syntax

```
Die Syntax des show job Statements ist
   show job jobid [ with WITHITEM {, WITHITEM} ]
   show job submittag = string [ with WITHITEM {, WITHITEM} ]
   WITHITEM:
        filter = ( FILTERITEM {, FILTERITEM} )
     recursive audit
   FILTERITEM:
        cancel
        change priority
        clear warning
        comment
        ignore named resource
        ignore resource
        ignore dependency [recursive]
        job in error
        kill
        renice
        rerun [ recursive ]
        restartable
        resume
        set exit state
        set resource state
        set state
        set warning
        submit [ suspend ]
        suspend
        timeout
        trigger failure
        trigger submit
```

unreachable

show job | 301

show job

Beschreibung

Beschreibung

Mit dem show job Statement bekommt man ausführliche Informationen über den spezifizierten Job. Der Job kann mittels seiner Id, oder aber, wenn beim Submit einen Submittag spezifiziert wurde, mittels des Submittags spezifiziert werden. Die Filter Option dient zum Selektieren von Audit Einträgen. Ohne Angabe der Filter Option werden alle Audit Einträge gezeigt. Ansonsten werden nur die Einträge der im Filter spezifizierten Typen ausgegeben.

Die **recursive audit** Option sammelt alle Audit Meldungen des gezeigten Jobs sowie die seiner direkten oder indirekten Children.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Die Nummer des Repository Objektes Der komplette Pfadname des Objektes Eigentümer des Objektes Der se_type ist der Objekttyp (JOB, BATCH
Eigentümer des Objektes
,
Der se_type ist der Objekttyp (JOB, BATCH
oder MILESTONE)
Die Run Program Zeile der Job Definition
Die Rerun Program Zeile der Job Definition
Die Kill Program Zeile der Job Definition
Die Workdir der Job Definition
Der Logfile der Job Definition
Gibt an ob das Logfile gekürzt werden soll bevor der Prozess startet oder ob die Loginfor- mation angehängt werden soll.
Der Error Logfile der Job Definition
Gibt an ob das Logfile gekürzt werden soll bevor der Prozess startet oder ob die Loginfor- mation angehängt werden soll.
Die erwartete Laufzeit der Job Definition
Priorität/Nice Value der Job Definition
Der Suspend Flag des Objektes
Der Master submittable Flag des Objektes

Fortsetzung auf der nächsten Seite

show job User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
SE_DEPENDENCY_MODE	Der Dependency Mode des Objektes
SE_ESP_NAME	Der Exit State Profile des Objektes
SE_ESM_NAME	Das Exit State Mapping der Job Definition
SE_ENV_NAME	Das Environment der Job Definition
SE_FP_NAME	Der Footprint der Job Definition
MASTER_ID	Die Objekt Id des Objektes der obersten Ebene in der Hierarchie dieses Objektes
CHILD_TAG	Tag zur ausschließlichen Erkennung von Jobs die mehrmals als Children des selben Jobs sub- mitted wurden
SE_VERSION	Die Ausführung von Definitionen die für dieses submitted Entity gültig sind
OWNER	Die Gruppe die Eigentümer des Objektes ist
PARENT_ID	Das Parent Objekt in der submission Hierarchie
SCOPE_ID	Die Id des Scopes
HTTPHOST	Der Hostname des Scopes für den Zugriff auf Logfiles via HTTP
HTTPPORT	Die HTTP Portnummer des Jobservers für den Zugriff auf Logfiles via HTTP
IS_STATIC	Flag das statische oder dynamische Submits von diesem Job anzeigt
MERGE_MODE	Zeigt an wie mehrere Submits von dem selben definierten Objekt in dem aktuellen Master Run gehandhabt werden
STATE	Der Status des Jobs, nicht zu verwechseln mit dem Exit State
IS_DISABLED	Zeigt an ob der Job bzw. Batch disabled submitted wurde.
IS_CANCELLED	Zeigt ein Cancel auf den Job durchgeführt wurde
JOB_ESD_ID	Der Exit State des Jobs nach der Ausführung
JOB_ESD_PREF	Die Präferenz zum mischen der Job Exit States mit den Child States
JOB_IS_FINAL	Flag das den finalen Status des Jobs anzeigt
JOB_IS_RESTARTABLE	Ein Flag das anzeigt das dieser Job Restartable ist
FINAL_ESD_ID	Der finale (merged) Exit Status des Objektes
Fortsetzung auf der nächsten S	eite

show job

Feld	Beschreibung
EXIT_CODE	Der Exit Code der letzten Kommandoaus
	führung
COMMANDLINE	Die erstellte Kommandozeile die bei der erste Ausführung genutzt wird
RR_COMMANDLINE	Erstellte Rerun Kommandozeile die bei der let zten Rerun Ausführung genutzt wird
WORKDIR	Der erstellte Workdir
LOGFILE	Das erstellte Logfile
ERRLOGFILE	Das erstellte Error Logfile
PID	Die Prozess ID des Controller Prozesses
EXT_PID	Die Prozess ID des User Prozesses
ERROR_MSG	Im Feld Error Message wird, falls ein Fehler be der Ausführung des Jobs im Time Schedulin aufgetreten ist, die übermittelte Fehlermeldun angezeigt. Ist kein Fehler aufgetreten, bleibt da Feld leer
KILL_ID	Die Submitted Entity Id des submitteten Ki Jobs
KILL_EXIT_CODE	Der Exit Code der letzten Kill Program Aus führung
IS_SUSPENDED	Flag das anzeigt ob das Objekt suspended ist
IS_SUSPENDED_LOCAL	Flag das anzeigt ob das Objekt lokal suspende ist (bei restart triggern mit suspend)
PRIORITY	Die aktuelle Priorität des Jobs
RAW_PRIORITY	Der uninterpretierte Prioritätswert des Jobs Anders als die Priorität, ist dieser Wert prak tisch nicht limitiert. Er wird benötigt um nac Manipulationen über Nice Profiles die korrekt Priorität wiederherstellen zu können.
NICEVALUE	Die aktuelle Nice Value des Jobs
NP_NICEVALUE	Der <i>np_nicevalue</i> ist der nice value, der als Folg von Aktivierungen (und Deaktivierungen) vo Nice Profiles entsteht.
MIN_PRIORITY	Minimale effektive Priorität die durch da natürliche Altern erreicht werden kann
AGING_AMOUNT	Die Anzahl Zeiteinheiten nach der die effektiv Priorität um 1 erhöht wird

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
AGING_BASE	Die Zeiteinheit die für das Alterungsintervall genutzt wird
DYNAMIC_PRIORITY	Die errechnete Priorität die aktuell vom System benutzt wird
PARENT_SUSPENDED	Die Anzahl der Parents die suspended wurden
SUBMIT_TS	Die Submit-Zeit
RESUME_TS	Der Zeitpunkt in dem der Job automatisch resumed wird
SYNC_TS	Der Zeitpunkt ab dem alle Abhängigkeiten erfüllt sind
RESOURCE_TS	Der Zeitpunkt ab dem alle Synchronizing Resources allokiert sind
RUNNABLE_TS	Der Zeitpunkt ab dem der Job in den Runnable Status wechselt
START_TS	Der Zeitpunkt des letzten Starts der Jobaus- führung
FINISH_TS	Der Endzeitpunkt der letzten Jobausführung
FINAL_TS	Der Zeitpunkt in dem das Objekt Final wird
CNT_SUBMITTED	Die Anzahl der Children im Submitted Status
CNT_DEPENDENCY_WAIT	Die Anzahl der Children im Dependency_Wait Status
CNT_SYNCHRONIZE_WAIT	Die Anzahl der Children im Synchronize_Wait Status
CNT_RESOURCE_WAIT	Die Anzahl der Children im Resource_Wait Status
CNT_RUNNABLE	Die Anzahl der Children im Runnable Status
CNT_STARTING	Die Anzahl der Children im Starting Status
CNT_STARTED	Die Anzahl der Children im Started Status
CNT_RUNNING	Die Anzahl der Children im Running Status
CNT_TO_KILL	Die Anzahl der Children im To_Kill Status
CNT_KILLED	Die Anzahl der Children im Killed Status
CNT_CANCELLED	Die Anzahl der Children im Cancelled Status
CNT_FINISHED	Die Anzahl der Children im Finished Status
CNT_FINAL	Die Anzahl der Children im Final Status
CNT_BROKEN_ACTIVE	Die Anzahl der Children im Broken_Active Status
CNT_BROKEN_FINISHED	Die Anzahl der Children im Broken_Finished Status
Fortsetzung auf der nächsten S	Seite

I 305

show job

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CNT_ERROR	Die Anzahl der Children im Error Status
CNT_RESTARTABLE	Die Anzahl der Children im Restartable Status
CNT_UNREACHABLE	Die Anzahl der Children im Unreachable Status
CNT_WARN	Anzahl der Children die Warnunmeldungen haben
WARN_COUNT	Anzahl der Warnmeldungen für das aktuelle Objekt
IDLE_TIME	
DEPENDENCY_WAIT_TIME	
SUSPEND_TIME	
SYNC_TIME	
RESOURCE_TIME	
JOBSERVER_TIME	
RESTARTABLE_TIME	
CHILD_WAIT_TIME	
PROCESS_TIME	
ACTIVE_TIME	
IDLE_PCT	
CHILDREN	Tabelle der Children
	Siehe auch Tabelle 22.25 auf Seite 308
PARENTS	Tabelle der Parents
	Siehe auch Tabelle 22.26 auf Seite 309
PARAMETER	Tabelle der Parameter
	Siehe auch Tabelle 22.27 auf Seite 309
REQUIRED_JOBS	Tabelle der benötigten Jobs
	Siehe auch Tabelle 22.28 auf Seite 311
DEPENDENT_JOBS	Tabelle der abhängigen Jobs
_	Siehe auch Tabelle 22.29 auf Seite 314
REQUIRED_RESOURCES	Tabelle der benötigten Resources
~ _	Siehe auch Tabelle 22.30 auf Seite 315
SUBMIT_PATH	Der Pfad vom Job bis zum Master über die Submit Hierarchie
IS_REPLACED	Flag das anzeigt ob der Job durch einen anderen ersetzt worden ist
TIMEOUT_AMOUNT	Die Zeit die der Job maximal auf seine Resource wartet
TIMEOUT_BASE	Die Zeit die der Job maximal auf seine Resource wartet
Fortsetzung auf der nächsten S	eite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TIMEOUT_STATE	Das Timeout des Scheduling Entities
RERUN_SEQ	Die Rerun-Reihenfolge
AUDIT_TRAIL	Tabelle der Protokoll Einträge
	Siehe auch Tabelle 22.31 auf Seite 316
CHILD_SUSPENDED	Die Anzahl der Children die Suspended wurden
CNT_PENDING	Die Anzahl der Children in einem Pending Status
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
SE_PRIVS	Die Privilegien auf den Scheduling Entitiy
SUBMITTAG	Einmaliger Marker der zur Submitzeit gegeben ist
UNRESOLVED_HANDLING	Bestimmt wie man in dem Fall das das benötigte Objekt nicht gefunden werden kann handeln soll
DEFINED_RESOURCES	Tabelle der Defined Resources des Objektes Siehe auch Tabelle 22.32 auf Seite 316

Tabelle 22.24.: Beschreibung der Output-Struktur des show job Statements

CHILDREN Das Layout der CHILDREN Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
CHILDID	Die Submitted Entity Id des Childs
CHILDPRIVS	Die Privilegien auf das Child-Objekt
CHILDSENAME	Der Name des Child-Objektes
CHILDSETYPE	Die Art des Child-Objektes
CHILDSEPRIVS	Die Privilegien auf das Child-Objekt
Fortsetzung auf der nächsten Seite	

User Commands show job

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
PARENTID	Die Id des Parents
PARENTPRIVS	Die Privilegien auf das Parent-Objekt
PARENTSENAME	Der Name des Parent-Objektes
PARENTSETYPE	Die Art des Parent-Objektes
PARENTSEPRIVS	Die Privilegien für die Job Definition die zu dem
	Parent gehört
IS_STATIC	Statischer Flag der hierarchy Definition
PRIORITY	Die Priorität auf die hierarchy Definition
SUSPEND	Der Suspend Modus der hierarchy Definition
MERGE_MODE	Der Merge Modus der hierarchy Definition
EST_NAME	Der Name der Exit State Translation der hierar-
	chy Definition
IGNORED_DEPENDENCIES	Ignored Dependencies Flag von der hierarchy
	Definition

Tabelle 22.25.: Output-Struktur der show job Subtabelle

PARENTS Das Layout der PARENTS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
CHILDID	Die Submitted Entity Id des Childs
CHILDPRIVS	Die Privilegien auf das Child-Objekt
CHILDSENAME	Der Name des Child-Objektes
CHILDSETYPE	Die Art des Child-Objektes
CHILDSEPRIVS	Die Privilegien auf das Child-Objekt
PARENTID	Die Id des Parents
PARENTPRIVS	Die Privilegien auf das Parent-Objekt
PARENTSENAME	Der Name des Parent-Objektes
PARENTSETYPE	Die Art des Parent-Objektes
PARENTSEPRIVS	Die Privilegien für die Job Definition die zu dem Parent gehört
IS_STATIC	Statischer Flag der hierarchy Definition
PRIORITY	Die Priorität auf die hierarchy Definition
SUSPEND	Der Suspend Modus der hierarchy Definition
MERGE_MODE	Der Merge Modus der hierarchy Definition
EST_NAME	Der Name der Exit State Translation der hierar- chy Definition

show job User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
IGNORED_DEPENDENCIES	Ignored Dependencies Flag von der hierarchy Definition

Tabelle 22.26.: Output-Struktur der show job Subtabelle

PARAMETER Das Layout der PARAMETER Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Parameters, der Variable oder des Ausdrucks
TYPE	Die Art des Parameters, der Variable oder des Ausdrucks
VALUE	Der Wert des Parameters, der Variable oder des Ausdrucks

Tabelle 22.27.: Output-Struktur der show job Subtabelle

REQUIRED_JOBS Das Layout der REQUIRED_JOBS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung	
ID	Die Nummer des Repository Objektes	
DEPENDENT_ID	Id des abhängigen Submitted Entities	
DEPENDENT_PATH	Der Pfad vom Job bis zum Master über die Submit Hierarchie	
DEPENDENT_PRIVS	Die Privilegien auf das abhängige Objekt	
DEPENDENT_ID_ORIG	Id des originalen abhängigen Submitted Entity auf dem die Abhängigkeit definiert ist für Ab- hängigkeiten die von den Parents geerbt wur- den	
DEPENDENT_PATH_ORIG	Der Pfad vom abhängigen Objekt bis zum Master über die Submit Hierarchie	
DEPENDENT_PRIVS_ORIG	Die Privilegien auf das originale abhängige Objekt	
DEPENDENCY_OPERATION	Definiert ob alle oder nur manche Ab- hängigkeiten des originalen Objektes erfüllt sein müssen	
Fortsetzung auf der nächsten S	eite	

show job

Feld	Beschreibung
REQUIRED_ID	Id des benötigten Submitted Entities
REQUIRED_PATH	Der Pfad vom benötigten Objekt bis zum Maste über die Submit Hierarchie
REQUIRED_PRIVS	Die Privilegien auf das benötigte Objekt
STATE	Der Status der Abhängigkeit (OPEN, FUL FILLED oder FAILED)
DD_ID	Id des Dependency Definition Objektes
DD_NAME	Name der Dependency Definition (deprecated)
DD_DEPENDENTNAME	Der komplette Pfadname des Objektes
DD_DEPENDENTTYPE	Die Art des Abhängigen Objektes
DD_DEPENDENTPRIVS	Privilegien auf das abhängige Objekt
DD_REQUIREDNAME	Pfadname der Definition des abhängigen Objek tes
DD_REQUIREDTYPE	Die Art des benötigten Objektes
DD_REQUIREDPRIVS	Die PrivilegienPrivileges on required objects definition
DD_UNRESOLVED_HANDLIN	GSpezifiziert wie man mit nicht auflösbaren Ab hängigkeiten beim Submit umgehen soll
DD_MODE	Gibt an ob nur der benötigte Job selbst oder der benötigte Job mit seinen Children Final sein muss
DD_STATES	Liste mit Exit States die das benötigte Objekt er reichen muss um die Abhängigkeit zu erfüllen
JOB_STATE	In der Liste Job State kann nach Jobs gefilter werden, welche den eingetragenen Job State haben
IS_SUSPENDED	Flag das anzeigt ob das Objekt suspended ist
PARENT_SUSPENDED	Job wird suspended weil sein Parent suspended ist
CNT_SUBMITTED	Die Anzahl der Children im Submitted Status
CNT_DEPENDENCY_WAIT	Die Anzahl der Children im Dependency_Wai Status
CNT_SYNCHRONIZE_WAIT	Die Anzahl der Children im Synchronize_Wai Status
CNT_RESOURCE_WAIT	Die Anzahl der Children im Resource_Wait Status
CNT_RUNNABLE	Die Anzahl der Children im Runnable Status

Fortsetzung der vorherigen S	Fortsetzung der vorherigen Seite		
Feld	Beschreibung		
CNT_STARTING	Die Anzahl der Children im Starting Status		
CNT_STARTED	Die Anzahl der Children im Started Status		
CNT_RUNNING	Die Anzahl der Children im Running Status		
CNT_TO_KILL	Die Anzahl der Children im To_Kill Status		
CNT_KILLED	Die Anzahl der Children im Killed Status		
CNT_CANCELLED	Die Anzahl der Children im Cancelled Status		
CNT_FINISHED	Die Anzahl der Children im Finished Status		
CNT_FINAL	Die Anzahl der Children im Final Status		
CNT_BROKEN_ACTIVE	Die Anzahl der Children im Broken_Active Sta-		
	tus		
CNT_BROKEN_FINISHED	Die Anzahl der Children im Broken_Finished		
	Status		
CNT_ERROR	Die Anzahl der Children im Error Status		
CNT_RESTARTABLE	Die Anzahl der Children im Restartable Status		
CNT_UNREACHABLE	Die Anzahl der Children im Unreachable Status		
JOB_IS_FINAL	Die Anzahl der Children im Is_Final Status		
CHILD_TAG	Tag zur ausschließlichen Erkennung von Jobs		
	die mehrmals als Children des selben Jobs sub-		
	mitted wurden		
FINAL_STATE	Der endgültige Status eines Jobs		
CHILDREN	Die Anzahl Kinder die der Job hat		
IGNORE	Flag das anzeigt ob die Resource Allocation ig-		
	noriert wird		
CHILD_SUSPENDED	Die Anzahl der Children die Suspended wur-		
	den		
CNT_PENDING	Die Anzahl der Children im Pending Status		
DD_CONDITION	Die Condition die zusätzlich erfüllt sein muss		
	damit die Abhängigkeit befriedigt ist		

Tabelle 22.28.: Output-Struktur der show job Subtabelle

DEPENDENT_JOBS Das Layout der DEPENDENT_JOBS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
Fortsetzung auf der nächsten	Seite

show job

Feld	Beschreibung
DEPENDENT_ID	Id des abhängigen Submitted Entities
DEPENDENT_PATH	Der Pfad vom Job bis zum Master über die Sub mit Hierarchie
DEPENDENT_PRIVS	Die Privilegien auf das abhängige Objekt
DEPENDENT_ID_ORIG	Id des originalen abhängigen Submitted Entity auf dem die Abhängigkeit definiert ist für Ab hängigkeiten die von den Parents geerbt wur den
DEPENDENT_PATH_ORIG	Der Pfad vom abhängigen Objekt bis zum Mas ter über die Submit Hierarchie
DEPENDENT_PRIVS_ORIG	Die Privilegien auf das originale abhängige Objekt
DEPENDENCY_OPERATION	Definiert ob alle oder nur manche Ab hängigkeiten des originalen Objektes erfüll sein müssen
REQUIRED_ID	Id des benötigten Submitted Entities
REQUIRED_PATH	Der Pfad vom benötigten Objekt bis zum Master über die Submit Hierarchie
REQUIRED_PRIVS	Die Privilegien auf das benötigte Objekt
STATE	Der Status der Abhängigkeit (OPEN, FUL FILLED oder FAILED)
DD_ID	Id des Dependency Definition Objektes
DD_NAME	Name der Dependency Definition (deprecated)
DD_DEPENDENTNAME	Der komplette Pfadname des Objektes
DD_DEPENDENTTYPE	Die Art des Abhängigen Objektes
DD_DEPENDENTPRIVS	Privilegien auf das abhängige Objekt
DD_REQUIREDNAME	Pfadname der Definition des abhängigen Objek tes
DD_REQUIREDTYPE	Die Art des benötigten Objektes
DD_REQUIREDPRIVS	Die PrivilegienPrivileges on required objects definition
DD_UNRESOLVED_HANDLIN	ICSpezifiziert wie man mit nicht auflösbaren Ab hängigkeiten beim Submit umgehen soll
DD_MODE	Gibt an ob nur der benötigte Job selbst ode der benötigte Job mit seinen Children Final sein muss

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
DD_STATES	Liste mit Exit States die das benötigte Objekt er-
	reichen muss um die Abhängigkeit zu erfüllen
JOB_STATE	In der Liste Job State kann nach Jobs gefiltert
	werden, welche den eingetragenen Job State
	haben
IS_SUSPENDED	Flag das anzeigt ob das Objekt suspended ist
PARENT_SUSPENDED	Job wird suspended weil sein Parent suspended
	ist
CNT_SUBMITTED	Die Anzahl der Children im Submitted Status
CNT_DEPENDENCY_WAIT	Die Anzahl der Children im Dependency_Wait Status
CNT_SYNCHRONIZE_WAIT	Die Anzahl der Children im Synchronize_Wait
	Status
CNT_RESOURCE_WAIT	Die Anzahl der Children im Resource_Wait Sta-
CNT DIDINIA DI E	tus
CNT_RUNNABLE	Die Anzahl der Children im Runnable Status
CNT_STARTING	Die Anzahl der Children im Starting Status
CNT_STARTED	Die Anzahl der Children im Started Status
CNT_RUNNING	Die Anzahl der Children im Running Status
CNT_TO_KILL	Die Anzahl der Children im To_Kill Status
CNT_KILLED	Die Anzahl der Children im Killed Status
CNT_CANCELLED	Die Anzahl der Children im Cancelled Status
CNT_FINISHED	Die Anzahl der Children im Finished Status
CNT_FINAL	Die Anzahl der Children im Final Status
CNT_BROKEN_ACTIVE	Die Anzahl der Children im Broken_Active Status
CNT_BROKEN_FINISHED	Die Anzahl der Children im Broken_Finished Status
CNT_ERROR	Die Anzahl der Children im Error Status
CNT_RESTARTABLE	Die Anzahl der Children im Restartable Status
CNT_UNREACHABLE	Die Anzahl der Children im Unreachable Status
JOB_IS_FINAL	Die Anzahl der Children im Is_Final Status
CHILD_TAG	Tag zur ausschließlichen Erkennung von Jobs
	die mehrmals als Children des selben Jobs submitted wurden
FINAL_STATE	Der endgültige Status eines Jobs
CHILDREN	Die Anzahl Kinder die der Job hat
IGNORE	Flag das anzeigt ob die Resource Allocation ig-
	noriert wird
Fortsetzung auf der nächsten S	Seite Seite

User Commands show job

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CHILD_SUSPENDED	Die Anzahl der Children die Suspended wurden
CNT_PENDING	Die Anzahl der Children im Pending Status
DD_CONDITION	Die Condition die zusätzlich erfüllt sein muss damit die Abhängigkeit befriedigt ist

Tabelle 22.29.: Output-Struktur der show job Subtabelle

REQUIRED_RESOURCES Das Layout der REQUIRED_RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung	
SCOPE_ID	Id des Scopes der die Resource allokiert hat	
SCOPE_NAME	Der vollqualifizierte Name des Scopes	
SCOPE_TYPE	Die Art des Scopes (SCOPE oder SERVER	
	FOLDER, BATCH oder JOB)	
SCOPE_PRIVS	Die Privilegien auf dem Scope	
RESOURCE_ID	Id der Required Resource	
RESOURCE_NAME	Kategorischer Pfadname der Requested Re	
	source	
RESOURCE_USAGE	Die Usage der Required Resource (STATIC	
	SYSTEM oder SYNCHRONIZING	
RESOURCE_OWNER	Name des Eigentümers der Requested Resource	
RESOURCE_PRIVS	Die Privilegien auf die Requested Resource	
RESOURCE_STATE	Der Status der Requested Resource	
RESOURCE_TIMESTAMP	Datetime of last time state was set on requested	
	resource	
REQUESTABLE_AMOUNT	Die Menge der Resourcen die maximal von einem Job angefordert werden darf	
TOTAL_AMOUNT	Die Menge die allokiert werden kann	
FREE_AMOUNT	Die freie Menge die angefordert werden kann	
REQUESTED_AMOUNT	Hierbei handelt es sich um die Anforderungs menge	
REQUESTED_LOCKMODE	Der beantragte Lockmode	
REQUESTED_STATES	Der beantragte Resource State	
RESERVED_AMOUNT	Die Menge die von der Requested Resource re serviert ist	

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ALLOCATED_AMOUNT	Die Menge die von der Requested Resource allokiert wurde
ALLOCATED_LOCKMODE	Der, von der Requested Resource, aktuell allokierte Lockmode
IGNORE	Flag das anzeigt ob die Resource Allocation ig- noriert wird
STICKY	Flag das anzeigt ob es eine Sticky Resource Allocation ist
STICKY_NAME	Optionaler Name der Sticky Anforderung
STICKY_PARENT	Parent Job innerhalb dessen die Sticky Anforderung gilt
STICKY_PARENT_TYPE	Type des Parents innerhalb dessen die Sticky Anforderung gilt
ONLINE	Flag das anzeigt ob die Resource für eine Allocation erhältlich ist
ALLOCATE_STATE	Der Status der Alocation (RESERVED, ALLO- CATED, AVAILABLE oder BLOCKED)
EXPIRE	Zeitpunkt der angibt wie alt eine Resource maximal bzw. minimal sein darf, je nachdem ob der Expire positiv oder negativ ist
EXPIRE_SIGN	Sign of the expiration condition, +/- indicating younger/older than
DEFINITION	Die Speicherstelle der Resource Definition

Tabelle 22.30.: Output-Struktur der show job Subtabelle

AUDIT_TRAIL Das Layout der AUDIT_TRAIL Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
USERNAME	Benutzername der diese Audit-Eingabe verursacht
TIME	Der Zeitpunkt in dem diese Audit-Eingabe erstellt wurde
TXID	Transaktionsnummer der Änderung
ACTION	Aktion die diese Audit-Eingabe verursacht
Fortsetzung auf der näch	sten Seite

User Commands show job

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ORIGINID	Die originale Objekt Id die diese Audit-Eingabe verursacht
JOBID	Dieses Feld wurde noch nicht beschrieben
JOBNAME	Dieses Feld wurde noch nicht beschrieben
COMMENT	Kommentar zum Objekt, wenn vorhanden
INFO	Zusätzliche Systeminformation über das Action
	Event das die Audit-Eingabe verursacht hat

Tabelle 22.31.: Output-Struktur der show job Subtabelle

DEFINED_RESOURCES Das Layout der DEFINED_RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Id der Defined Resource
RESOURCE_NAME	Kompletter Pfadname des Defined Objektes
RESOURCE_USAGE	Die Usage der Resource (STATIC, SYSTEM oder SYNCHRONIZING)
RESOURCE_OWNER	Der Eigentümer der Resource
RESOURCE_PRIVS	Die Privilegien auf die Resource
RESOURCE_STATE	Der aktuelle Status der Resource
RESOURCE_TIMESTAMP	Der letzte Zeitpunkt das der Resource Status dieser Resource gesetzt wurde
REQUESTABLE_AMOUNT	Die Menge der Resourcen die maximal von einem Job angefordert werden darf
TOTAL_AMOUNT	Der komplette Amount der allokiert werden kann
FREE_AMOUNT	Die freie Menge die allokiert werden darf
ONLINE	Zeigt an ob die Resource allokiert werden kann oder nicht

Tabelle 22.32.: Output-Struktur der show job Subtabelle

show job definition

Zweck

Das show job definition Statement wird eingesetzt um detaillierte Informationen Zweck über die definierte Job Definition zu bekommen.

Syntax

Die Syntax des show job definition Statements ist

Syntax

show job definition folderpath

Beschreibung

Mit dem show job definition Statement bekommt man ausführliche Informationen über die spezifizierte Job Definition.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der vollständige Pfadname der Job Definition
OWNER	Die Gruppe die Eigentümer des Objektes ist
TYPE	Der Objekttyp (Batch, Job oder Milestone)
INHERIT_PRIVS	Vom übergeordneten Ordner zu erbende Privilegien
RUN_PROGRAM	Die Kommandozeile zur Ausführung eines Jobs der zum ersten mal ausgeführt wird
RERUN_PROGRAM	Die Kommandozeile um einen sumbmitteten Job neu zu starten
KILL_PROGRAM	Die Kommandozeile zur Ausführung der Löschung eines laufenden Jobs
WORKDIR	Das Working Directory von wo aus das Kommando ausgeführt werden soll

show job definition

Fortsetzung der vorherigen S	
Feld	Beschreibung
LOGFILE	Die Datei in die der stdout des Kommandos geschrieben wird
TRUNC_LOG	Zeigt an ob das Log überschrieben oder angehängt wird
ERRLOGFILE	Die Datei in die der stderr des Kommandos geschrieben wird
TRUNC_ERRLOG	Zeigt an ob das Fehler Log überschrieben oder angehängt wird
EXPECTED_RUNTIME	Die geschätzte Laufzeit in Sekunden. Diese kann in Triggern zur Laufzeitüberwachung herangezogen werden
EXPECTED_FINALTIME	Die geschätzte Laufzeit bis zum Final Sta- tus in Sekunden. Diese kann in Triggern zur Laufzeitüberwachung herangezogen werden
PRIORITY	Priorität / Nice Value des Jobs oder Batches
MIN_PRIORITY	Minimale effektive Priorität die durch das natürliche Altern erreicht werden kann
AGING_AMOUNT	Die Anzahl Zeiteinheiten nach der die effektive Priorität um 1 erhöht wird
AGING_BASE	Die Zeiteinheit die für das Alterungsintervall genutzt wird
SUBMIT_SUSPENDED	Flag das angibt ob das Objekt nach dem Submit suspended werden soll
RESUME_AT	Falls der Job suspended submitted werden soll, erfolgt ein automatischer Resume zum angegebenen Zeitpunkt
RESUME_IN	Falls der Job suspended submitted werden soll, erfolgt ein automatischer Resume nach der angegebenen Anzahl Zeiteinheiten
RESUME_BASE	Zeiteinheitsangabe für RESUME_IN
MASTER_SUBMITTABLE	Flag das angibt ob der Job oder Batch als stan- dalone Master submitted werden kann
TIMEOUT_AMOUNT	Die Anzahl Zeiteinheiten die gewartet wird bis das Timeout eintritt
TIMEOUT_BASE	Die Einheit die genutzt wird um das Timeout in Sekunden, Minuten, Stunden oder Tagen zu spezifizieren

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TIMEOUT_STATE	Exit Status der für ein Job gesetzt wird wenn der Timeout eintritt
DEPENDENCY_MODE	Zeigt an ob alle oder nur eine Abhängigkeit erfüllt sein müssen um den Job zu starten
ESP_NAME	Name des Exit State Profiles des Objektes
ESM_NAME	Name des Exit State Mappings das für diesen Job genutzt wird
ENV_NAME	Name des Environments das der Job anfordert
FP_NAME	Name des Footprints den der Job benutzt
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
CHILDREN	Tabelle der Children
	Siehe auch Tabelle 22.34 auf Seite 321
PARENTS	Tabelle der Parents
	Siehe auch Tabelle 22.35 auf Seite 322
PARAMETER	Tabelle der Parameter und Variable die für dieses Objekt definiert sind
	Siehe auch Tabelle 22.36 auf Seite 323
REFERENCES	Tabelle der Parameter Referenzen auf dieses Objekt
REQUIRED_JOBS	Tabelle der Objekte von denen die nachfolgenden Objekte abhängig sind
	Siehe auch Tabelle 22.37 auf Seite 324
DEPENDENT_JOBS	Tabelle der Objekte die von den nachfolgenden Objekten abhängig sind
	Siehe auch Tabelle 22.38 auf Seite 325
REQUIRED_RESOURCES	Tabelle der Resource Anforderungen die nicht
	im Environent und Footprint enthalten sind
	Siehe auch Tabelle 22.39 auf Seite 326
DEFINED_RESOURCES	Tabelle von Resourcen zum instanziieren in die
	submit Zeit, sichtbar um Children zu submitten
Fortsetzung auf der nächster	n Seite

User Commands show job definition

Fortsetzung der vorherigen Seite		
Feld	Beschreibung	

Tabelle 22.33.: Beschreibung der Output-Struktur des show job definition Statements

CHILDREN Das Layout der CHILDREN Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung	
ID	Die Nummer des Repository Objektes	
CHILDNAME	Kompletter Pfadname des Child Objektes	
CHILDTYPE	Der Child Typ (JOB, BATCH oder MILESTONE	
CHILDPRIVS	Ein String der die Benutzerprivilegien des Child Objektes enthält	
PARENTNAME	Kompletter Pfadname des Parent Objektes	
PARENTTYPE	Der Parent Typ (JOB, BATCH oder MILE STONE)	
PARENTPRIVS	Ein String der die Benutzerprivilegien des Par ent Objektes enthält	
ALIAS_NAME	Name zum referieren auf Child Definitionen bei dynamischen Submits	
IS_STATIC	Der is_static Flag gibt an ob der Job statisch ode dynamisch submitted werden soll	
IS_DISABLED	Flag das angibt ob das Child ausgeführt oder übersprungen wird	
PRIORITY	Der Nicevalue welcher der Childrens zugefüß wurde	
SUSPEND	Bestimmt ob das Child beim Submit suspended werden soll	
RESUME_AT	Falls der Job suspended submitted werden soll, erfolgt ein automatischer Resume zum angegebenen Zeitpunkt	
RESUME_IN	Falls der Job suspended submitted werden soll erfolgt ein automatischer Resume nach der angegebenen Anzahl Zeiteinheiten	
RESUME_BASE	Zeiteinheitsangabe für RESUME_IN	
MERGE_MODE	Legt fest wie die Kondition das selbe Objekt be handelt das mehr als einmal in der submissior Hierarchie auftritt	

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
EST_NAME	Eine Exit State Translation die benutzt wird um die Exit States der Children nach den Exit States der Parents zu Übersetzen
IGNORED_DEPENDENCIES	Liste mit den Namen der Abhängigkeiten zum Ignorieren der Abhängigkeiten der Parents

Tabelle 22.34.: Output-Struktur der show job definition Subtabelle

PARENTS Das Layout der PARENTS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung	
ID	Die Nummer des Repository Objektes	
CHILDNAME	Kompletter Pfadname des Child Objektes	
CHILDTYPE	Der Child Typ (JOB, BATCH oder MILESTONE)	
CHILDPRIVS	Ein String der die Benutzerprivilegien des Child Objektes enthält	
PARENTNAME	Kompletter Pfadname des Parent Objektes	
PARENTTYPE	Der Parent Typ (JOB, BATCH oder MILE-STONE)	
PARENTPRIVS	Ein String der die Benutzerprivilegien des Parent Objektes enthält	
ALIAS_NAME	Name zum referieren auf Child Definitionen bei dynamischen Submits	
IS_STATIC	Der is_static Flag gibt an ob der Job statisch oder dynamisch submitted werden soll	
IS_DISABLED	Flag das angibt ob das Child ausgeführt oder übersprungen wird	
PRIORITY	Der Nicevalue welcher der Childrens zugefügt wurde	
SUSPEND	Bestimmt ob das Child beim Submit suspended werden soll	
RESUME_AT	Falls der Job suspended submitted werden soll, erfolgt ein automatischer Resume zum angegebenen Zeitpunkt	
RESUME_IN	Falls der Job suspended submitted werden soll, erfolgt ein automatischer Resume nach der angegebenen Anzahl Zeiteinheiten	
Fortsetzung auf der näc	Fortsetzung auf der nächsten Seite	

User Commands show job definition

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
RESUME_BASE	Zeiteinheitsangabe für RESUME_IN
MERGE_MODE	Legt fest wie die Kondition das selbe Objekt be-
	handelt das mehr als einmal in der submission
	Hierarchie auftritt
EST_NAME	Eine Exit State Translation die benutzt wird um
	die Exit States der Children nach den Exit States
	der Parents zu Übersetzen
IGNORED_DEPENDENCIES	Liste mit den Namen der Abhängigkeiten zum
	Ignorieren der Abhängigkeiten der Parents

Tabelle 22.35.: Output-Struktur der show job definition Subtabelle

PARAMETER Das Layout der PARAMETER Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
EXPORT_NAME	Der Export Name definiert den Namen unter dem der Wert des Parameters in die Prozes- sumgebung exportiert wird.
ТҮРЕ	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
EXPRESSION	Expression für den Parameter des Typs EX- PRESSION
DEFAULT_VALUE	Der Default Value des Parameters
REFERENCE_TYPE	Typ des Objektes auf das referenziert wird
REFERENCE_PATH	Der Pfad des Objektes auf das referenziert wird
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes auf das referenziert wird
REFERENCE_PARAMETER	Name des Parameters auf den referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
Fortsetzung auf der nächsten	Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ТҮРЕ	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
REFERENCE_TYPE	Typ des Objektes welches den Parameter referenziert
REFERENCE_PATH	Der Pfad des Objektes welches den Parameter referenziert
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes welches den Parameter referenziert
REFERENCE_PARAMETER	Name des Parameters auf dem referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.36.: Output-Struktur der show job definition Subtabelle

REQUIRED_JOBS Das Layout der REQUIRED_JOBS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
DEPENDENTNAME	Der komplette Pfadname des abhängigen Objektes
DEPENDENTTYPE	Der Typ des abhängigen Objektes (JOB, BATCH oder MILESTONE)
DEPENDENTPRIVS	String der die Benutzerprivilegien des abhängigen Objektes enthält
REQUIREDNAME	Der komplette Pfadname des benötigten Objektes
REQUIREDTYPE	Der Typ des benötigten Objektes (JOB, BATCH oder MILESTONE)
REQUIREDPRIVS	String der die Benutzerprivilegien des benötigten Objektes enthält
UNRESOLVED_HANDLING	Bestimmt wie man in dem Fall dass das benötigte Objekt nicht gefunden werden kann handeln soll
Fortsetzung auf der nächsten Seite	

show job definition

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
MODE	Definiert ob Abhängigkeiten bei einem finalen Job Status (JOB_FINAL) als erfüllt behandelt werden oder final States von Objekten die Child Objekte beinhalten (ALL_FINAL)
STATE_SELECTION	Die State Selection gibt an, wie die benötigten Exit States ermittelt werden. Es gibt die Optionen FINAL, ALL_REACHABLE, UNREACHABLE und DEFAULT. Im Falle von FINAL können die benötigte Exit States expliziert aufgeführt sein
CONDITION	Die zusätzlichen Konditionen müssen erfüllt sein
STATES	Kommas trennen Listen von zugelassenen Exit States die das benötigte Objekt erreichen muss um die Abhängigkeiten zu erfüllen. Wenn eine zusätzliche Kondition spezifiziert wurde, wird es nach dem Doppelpunkt angezeigt

Tabelle 22.37.: Output-Struktur der show job definition Subtabelle

DEPENDENT_JOBS Das Layout der DEPENDENT_JOBS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
DEPENDENTNAME	Der komplette Pfadname des abhängigen Objektes
DEPENDENTTYPE	Der Typ des abhängigen Objektes (JOB, BATCH oder MILESTONE)
DEPENDENTPRIVS	String der die Benutzerprivilegien des abhängigen Objektes enthält
REQUIREDNAME	Der komplette Pfadname des benötigten Objektes
REQUIREDTYPE	Der Typ des benötigten Objektes (JOB, BATCH oder MILESTONE)
Fortsetzung auf der nächsten Seite	

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
REQUIREDPRIVS	String der die Benutzerprivilegien des benötigten Objektes enthält
UNRESOLVED_HANDLING	Bestimmt wie man in dem Fall dass das benötigte Objekt nicht gefunden werden kann handeln soll
MODE	Definiert ob Abhängigkeiten bei einem finalen Job Status (JOB_FINAL) als erfüllt behandelt werden oder final States von Objekten die Child Objekte beinhalten (ALL_FINAL)
STATE_SELECTION	Die State Selection gibt an, wie die benötigten Exit States ermittelt werden. Es gibt die Optionen FINAL, ALL_REACHABLE, UNREACHABLE und DEFAULT. Im Falle von FINAL können die benötigte Exit States expliziert aufgeführt sein
CONDITION	Die zusätzlichen Konditionen müssen erfüllt sein
STATES	Kommas trennen Listen von zugelassenen Exit States die das benötigte Objekt erreichen muss um die Abhängigkeiten zu erfüllen. Wenn eine zusätzliche Kondition spezifiziert wurde, wird es nach dem Doppelpunkt angezeigt

Tabelle 22.38.: Output-Struktur der show job definition Subtabelle

$\begin{tabular}{ll} \textbf{REQUIRED_RESOURCES} & Das\ Layout\ der\ REQUIRED_RESOURCES\ Tabelle\ wird\ in\ untenstehender\ Tabelle\ gezeigt. \end{tabular}$

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
RESOURCE_NAME	Kompletter Pfadname der benötigten Named Resource
RESOURCE_USAGE	Die Anwendung der benötigten Resource (STATIC, SYSTEM oder SYNCHRONIZING)
RESOURCE_PRIVS	String der die Benutzerprivilegien der Named Resource beinhaltet
AMOUNT	Die benötigte Menge bei System oder Synchro- nizing Resources
Fortsetzung auf der nächsten S	Seite

show job definition

Fortsetzung der vorherigen Seit	Fortsetzung der vorherigen Seite	
Feld	Beschreibung	
KEEP_MODE	Zeigt an ob die Resource beim Jobende (NO-KEEP), finalen Job Status (KEEP) oder beim finalen Job Status freigegeben werden soll, einschließlich aller Children (KEEP_FINAL)	
IS_STICKY	Zeigt an ob die Resource Zuordnung für nachfolgende Jobs beibehalten wird	
STICKY_NAME	Optionaler Name der Sticky Anforderung	
STICKY_PARENT	Parent Job Definition innerhalb deren die Sticky Anforderung gilt	
RESOURCE_STATE_MAPPING	Das Resource State Mapping gibt an wie und ob der State der Resource nach Beendigung des Jobs geändert werden soll	
EXPIRED_AMOUNT	Die maximale Anzahl von Einheiten, ausgedrückt durch EXPIRED_BASE, die diese Resource berühren kann bevor sie eine Allokation dieser Synchronizing Resource ermöglicht	
EXPIRED_BASE	Die Einheit um den Ablauf zu spezifizieren MINUTEN, STUNDEN, TAGE, WOCHEN, MONATE und JAHRE	
LOCKMODE	Der Sperrmodus zum allokieren von Synchronizing Resourcen (N, S, SX, X)	
STATES	Kommas trennen Listen von zugelassenen Exit States die das benötigte Objekt erreichen muss um die Abhängigkeiten zu erfüllen	
DEFINITION	Der Ursprung der Resource Anforderung (RE-QUIREMENT, FOOTPRINT, FOLDER oder EN-VIRONMENT)	
ORIGIN	Name der Resource Anforderungs Definition, ungültig im Falle einer vollständigen An- forderung	
CONDITION	Die optionale Condition die für Anforderungen von static Resources definiert sein darf	

Tabelle 22.39.: Output-Struktur der show job definition Subtabelle

show named resource

Zweck

Das show named resource Statement wird eingesetzt um detaillierte Informatio- Zweck nen über die Named Resource zu bekommen.

Syntax

Die Syntax des show named resource Statements ist

Syntax

show named resource resourcepath [with EXPAND]

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Mit dem show named resource Statement bekommt man ausführliche Informationen über die Named Resource.

Beschreibung

expand Da die Anzahl Job Definitions in der Tabelle JOB_DEFINITIONS sehr groß werden kann, werden sie per Default nicht angezeigt. Wenn die Option **expand = all** benutzt wird, werden alle Job Definitions sowie die Folder in die sie liegen samt Folderhierarchie ausgegeben. Durch die Spezifikation einzelner (Folder) Ids, können einzelne Pfade der Hierarchie selektiert werden.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name der Named Resource
OWNER	Eigentümer der Named Resource
USAGE	Die Usage gibt an um welchen Typ Resource es
	sich handelt
Fortsetzung auf der nächsten	Seite

User Commands show named resource

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
INHERIT_PRIVS	Vom übergeordneten Ordner zu erbende Privi-
	legien
RESOURCE_STATE_PROFILE	Es handelt sich hier um das zur Resource zuge- ordnete Resource State Profile
FACTOR	Der Faktor mit dem Allokierungen multipliziert werden
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
RESOURCES	Das sind die Instanzen der Named Resource
	Siehe auch Tabelle 22.41 auf Seite 329
PARAMETERS	Das sind die definierten Parameter der Named
	Resource
	Siehe auch Tabelle 22.42 auf Seite 329
JOB_DEFINITIONS	Das sind die Job Definitions die die Named Re-
	source anfordern
	Siehe auch Tabelle 22.43 auf Seite 330

Tabelle 22.40.: Beschreibung der Output-Struktur des show named resource Statements

RESOURCES Das Layout der RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
SCOPE	Hier erscheinen die Namen der Scopes, Sub-
	mitted Entities, Scheduling Entities der Folder,
	welche die jeweilige Named Resource anbieten
Fortsetzung auf der nächsten S	Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TYPE	Hierbei handelt es sich um den Resource-Typ
OWNER	Die Gruppe die Eigentümer des Objektes ist
STATE	Zeigt den Status der Resource an
REQUESTABLE_AMOUNT	Die Menge der Resourcen die maximal von einem Job angefordert werden darf
AMOUNT	Der Amount gibt die aktuelle Anzahl von Instanzen der Named Resource für diesen Scope oder Jobserver an
FREE_AMOUNT	Der Free Amount bezeichnet die Anzahl aller noch nicht von Jobs belegten Instanzen einer Resource innerhalb des gewählten Scopes oder Jobservers
IS_ONLINE	Zeigt an ob die Resource Online ist oder nicht
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.41.: Output-Struktur der show named resource Subtabelle

PARAMETERS Das Layout der PARAMETERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Parameters
ТҮРЕ	Beim Type handelt es sich um den Typ des Parameters. Local oder Local Constant
DEFAULT_VALUE	Beim Default Value unterscheiden wir zwischen Constants und Local Constants. Bei Constants ist er der Wert des Parameters und bei Local Constants der Default Wert
TAG	Dieses Feld wurde noch nicht beschrieben
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.42.: Output-Struktur der show named resource Subtabelle

JOB_DEFINITIONS Das Layout der JOB_DEFINITIONS Tabelle wird in untenstehender Tabelle gezeigt.

User Commands show named resource

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name der Job Definition
AMOUNT	Die Resource-Menge die von dem Job benötigt wird
KEEP_MODE	Das ist der Wert des Keep Parameters für die Resource-Anforderung des Jobs
IS_STICKY	Gibt an ob es Sticky Request ist oder nicht
STICKY_NAME	Optionaler Name der Sticky Anforderung
STICKY_PARENT	Parent Job Definition innerhalb deren die Sticky Anforderung gilt
RESOURCE_STATE_MAPPING	Wurde bei der Resource-Anforderung ein Resource State Mapping angegeben, wird dies hier angezeigt
EXPIRED_AMOUNT	Die Anzahl der Einheiten. Wenn der Expired Amount positiv ist, bedeutet das, dass der Sta- tuswechsel maximal die angegebene Zeit her sein darf. Ist der Amount negativ, muss es min- imal so lange her sein
EXPIRED_BASE	Die Einheit in Minuten, Stunden, Tage, Wochen, Monate und Jahre
LOCKMODE	Der Lockmode beschreibt den Modus des Zugriffs auf diese Resource (exclusiv, shared etc.
STATES	Falls mehrere States für diesen Job akzeptabel sind, werden die einzelnen States durch Komma getrennt
CONDITION	Die Condition die für Anforderungen von Static Resources definiert sein darf
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.43.: Output-Struktur der show named resource Subtabelle

show resource

User Commands

show resource

Zweck

Das show resource Statement wird eingesetzt um detaillierte Informationen über Zweck die Resource zu bekommen.

Syntax

Die Syntax des show resource Statements ist

Syntax

show RESOURCE_URL

RESOURCE_URL:

resource resourcepath in folderpath resource resourcepath in serverpath

Beschreibung

Mit dem show resource Statement bekommt man ausführliche Informationen Beschreibung über die Resource.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name der Resource
SCOPENAME	Name des Scopes in dem der Pool angelegt wurde
OWNER	Die Gruppe die Eigentümer des Objektes ist
LINK_ID	Id der Ressource auf die verwiesen wird
LINK_SCOPE	Scopename der Ressource auf die verwiesen wird
BASE_ID	Id der Ressource auf die letztendlich verwiesen wird
Fortsetzung auf der nächsten S	eite

show resource

Fortsetzung der vorherigen Sei Feld	Beschreibung
BASE_SCOPE	Scopename der Ressource auf die letztendlich verwiesen wird
MANAGER_ID	Id des Managing Pools
MANAGER_NAME	Name des Managing Pools
MANAGER_SCOPENAME	Name des Scopes in dem der managing Pool ar gelegt wurde
USAGE	Die Usage gibt an um welchen Typ Resource e sich handelt
RESOURCE_STATE_PROFILE	Hier handelt es sich um das zur Resource zuge ordnete Resource State Profile
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
TAG	Der Tag ist ein optionaler Kurzname für die Resource
STATE	Bei dem State handelt es sich um den aktuelle Status der Resource in diesem Scope oder Job server
TIMESTAMP	Der Timestamp gibt die Zeit des letzten Statuswechsels einer Resource an
REQUESTABLE_AMOUNT	Der Requestable Amount ist die maximal Menge die angefordert werden kann
DEFINED_AMOUNT	Die Menge die vorhanden ist wenn die Resourc nicht pooled ist
AMOUNT	Die tatsächliche Menge die vorhanden ist
FREE_AMOUNT	Der Free Amount bezeichnet die Anzahl alle noch nicht von Jobs belegten Instanzen einer Re source
IS_ONLINE	Is Online ist ein Indikator der angibt ob die Resource online ist oder nicht
FACTOR	Dies ist der Korrekturfaktor mit dem ange forderte Amounts multipliziert werden
TRACE_INTERVAL	Das trace Interval ist die minimale Zeit zwis chen dem Scheiben von Trace Records in Sekur den
TRACE_BASE	Die trace Base ist es die Basis für den Auswertungszeitraum
TRACE_BASE_MULTIPLIER	Der base Multiplier bestimmt den Multiplika tionsfaktor von der trace Base

show resource User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
TD0_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^0$ Sekunden
TD1_AVG	Die durchschnittliche Resourcebelegung der letzten $B * M^1$ Sekunden
TD2_AVG	Die durchschnittliche Resourcebelegung der letzten $B\ast M^2$ Sekunden
LW_AVG	Die durchschnittliche Resourcebelegung seit dem letzten Schreiben eines Trace Records
LAST_WRITE	Zeitpunkt des letzen Schreibens eines Trace Records
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
ALLOCATIONS	Das ist eine Tabelle mit Resource Belegungen Siehe auch Tabelle 22.45 auf Seite 334
PARAMETERS	Im Tab Parameters können zusätzliche Informationen zu einer Resource gespeichert werden Siehe auch Tabelle 22.46 auf Seite 335

Tabelle 22.44.: Beschreibung der Output-Struktur des show resource Statements

ALLOCATIONS Das Layout der ALLOCATIONS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
JOBID	Hierbei handelt es sich um die ID der Jobin- stanz, welche durch ein direktes Submit des Jobs oder durch ein Submit des Master Batches oder Jobs gestartet wurde
Fortsetzung auf der nächsten	Seite

show resource

Fortsetzung der vorherigen S Feld	Beschreibung
MASTERID	Hierbei handelt es sich um die ID der Job- oder Batchinstanz, welche als Master Job gestartet wurde und den aktuellen Job als Child beinhaltet
JOBTYPE	Hierbei handelt es sich um den Typ des Jobs
JOBNAME	Hierbei handelt es sich um den Namen des Jobs
AMOUNT	Der Amount ist die Menge die Vorhanden ist
KEEP_MODE	Der Keep Parameter gibt an ob der Job die aktuelle Resource "Sticky" hält oder nicht. Es gibt folgende Ausführungen: KEEP, NO KEEP und KEEP FINAL
IS_STICKY	Die Resource wird nur freigegeben wenn im gleichen Batch keine weiteren Sticky Requests für diese Named Resource vorhanden sind
STICKY_NAME	Optionaler Name der Sticky Anforderung
STICKY_PARENT	Parent Job innerhalb dessen die Sticky Anforderung gilt
STICKY_PARENT_TYPE	Type des Parents innerhalb dessen die Sticky Anforderung gilt
LOCKMODE	Der Lockmode gibt an mit welchem Zugriffs- modus die Resource vom aktuellen Job belegt wird
RSM_NAME	Der Name des Resource State Mappings
ТҮРЕ	Die Art der Allokierung: Available, Blocked, Allokations, Master_Reservation, Reservation
TYPESORT	Hilfe für die Sortierung der Allocations
P	Die Priorität des Jobs
EP	Die effektive Priorität des Jobs
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.45.: Output-Struktur der show resource Subtabelle

PARAMETERS Das Layout der PARAMETERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
EXPORT_NAME	Der Export Name definiert den Namen unter dem der Wert des Parameters in die Prozes- sumgebung exportiert wird.
TYPE	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
EXPRESSION	Expression für den Parameter des Typs EX- PRESSION
DEFAULT_VALUE	Der Default Value des Parameters
REFERENCE_TYPE	Typ des Objektes auf das referenziert wird
REFERENCE_PATH	Der Pfad des Objektes auf das referenziert wird
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes auf das referenziert wird
REFERENCE_PARAMETER	Name des Parameters auf den referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
TYPE	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
REFERENCE_TYPE	Typ des Objektes welches den Parameter referenziert
REFERENCE_PATH	Der Pfad des Objektes welches den Parameter referenziert
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes welches den Parameter referenziert
REFERENCE_PARAMETER	Name des Parameters auf dem referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.46.: Output-Struktur der show resource Subtabelle

show resource state definition

show resource state definition

Zweck

Zweck

Der Zweck der show resource state definition ist, detaillierte Informationen über die spezifizierte Resource State Definition zu bekommen.

Syntax

Syntax

Die Syntax des show resource state definition Statements ist

show resource state definition statename

Beschreibung

Beschreibung

Mit dem show resource state definition Statement bekommt man ausführliche Informationen über die Resource State Definition.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt
	hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 22.47.: Beschreibung der Output-Struktur des show resource state definition Statements

show resource state mapping

Zweck

Das show resource state mapping Statement wird eingesetzt um detaillierte In- Zweck formationen über das spezifizierte Mapping zu bekommen.

Syntax

Die Syntax des show resource state mapping Statements ist

Syntax

show resource state mapping profilename

Beschreibung

Mit dem show resource state mapping Statement bekommt man ausführliche Beschreibung Informationen über das spezifizierte Mapping.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Resource State Mappings
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
MAPPINGS	Eine Tabelle mit Übersetzungen von Exit State nach Resource State

User Commands show resource state mapping

Fortsetzung der von	herigen Seite
Feld	Beschreibung
	Siehe auch Tabelle 22.49 auf Seite 338

Tabelle 22.48.: Beschreibung der Output-Struktur des show resource state mapping Statements

MAPPINGS Das Layout der MAPPINGS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ESD_NAME	Name der Exit State Definition
RSD_FROM	Der ursprüngliche Status der Resource
RSD_TO	Der aktuelle Status der Resource

Tabelle 22.49.: Output-Struktur der show resource state mapping Subtabelle

show resource state profile

Zweck

Der Zweck des show resource state profile Statements ist, detaillierte Informa- Zweck tionen über die spezifizierten Resorce State Profiles zu bekommen.

Syntax

Die Syntax des show resource state profile Statements ist

Syntax

show resource state profile profilename

Beschreibung

Mit dem show resource state profile Statement bekommt man ausführliche Informationen über die spezifizierten Resource State Profiles.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
INITIAL_STATE	Dieses Feld definiert den initialen Status der Resource. Dieser Resource State muss nicht in der Liste gültiger Resource States vorhanden sein
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

User Commands show resource state profile

Fortsetzung der vor	herigen Seite
Feld	Beschreibung
STATES	In der Tabelle States stehen in der Spalte Re-
	source State die gültigen Resource States
	Siehe auch Tabelle 22.51 auf Seite 340

Tabelle 22.50.: Beschreibung der Output-Struktur des show resource state profile Statements

STATES Das Layout der STATES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
RSD_NAME	Name der Resource State Definition
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.51.: Output-Struktur der show resource state profile Subtabelle

show schedule

User Commands

show schedule

Zweck

Das show schedule Statement wird eingesetzt um detaillierte Informationen über Zweck den spezifizierten Zeitplan zu erhalten.

Syntax

Die Syntax des show schedule Statements ist

Syntax

show schedule *schedulepath*

Beschreibung

Mit dem show schedule Statement bekommt man ausführliche Informationen Beschreibung über den spezifizierten Zeitplan.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
INHERIT_PRIVS	Vom übergeordneten Ordner zu erbende Privilegien
INTERVAL	Hierbei handelt es sich um Blöcke bestimmter Längen im Time Scheduling
TIME_ZONE	Die Zeitzone in der der Schedule gerechnet werden soll
ACTIVE	Nur wenn das Flag gesetzt ist, werden die Jobs submitted
EFF_ACTIVE	Gibt an ob ein Parent vom Scheduler auf Inaktiv gesetzt ist
CREATOR	Name des Benutzers der dieses Objekt angelegt hat

User Commands show schedule

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.52.: Beschreibung der Output-Struktur des show schedule Statements

show scheduled event

Zweck

Der Zweck des show scheduled event Statements ist, detaillierte Informationen Zweck über das spezifizierte Event zu bekommen.

Syntax

Die Syntax des show scheduled event Statements ist

Syntax

show scheduled event schedulepath . eventname

Beschreibung

Mit dem show scheduled event Statement bekommt man ausführliche Informa- Be tionen über das spezifizierte Event.

Beschreibung

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
SCHEDULE	Der Schedule definiert den Zeitplan
EVENT	Der Event beschreibt was ausgeführt werden soll
ACTIVE	Nur wenn das Flag gesetzt ist, werden die Jobs submitted
EFF_ACTIVE	Gibt an ob ein Parent vom Scheduler auf Inaktiv gesetzt ist
BROKEN	Mittels des Broken Feldes kann geprüft werden, ob beim Submit des Jobs ein Fehler aufgetreten ist
ERROR_CODE	Im Feld Error Code wird, falls ein Fehler bei der Ausführung des Jobs im Time Schedul- ing aufgetreten ist, der übermittelte Fehlercode angezeigt. Ist kein Fehler aufgetreten, bleibt das Feld leer

show scheduled event

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ERROR_MSG	Im Feld Error Message wird, falls ein Fehler bei der Ausführung des Jobs im Time Scheduling aufgetreten ist, die übermittelte Fehlermeldung angezeigt. Ist kein Fehler aufgetreten, bleibt das Feld leer
LAST_START	Hier wird der letzte Ausführungszeitpunkt des Jobs durch das Scheduling System angezeigt
NEXT_START	Hier wird der nächste geplante Ausführungszeitpunkt des Tasks durch das Scheduling System angezeigt
NEXT_CALC	Wenn der Next Start leer ist gibt der Next Clac den Zeitpunkt an wann nach einem nächsten Startzeitpunkt gesucht wird. Ansonsten findet die neue Berechnung zum Next Start Zeitpunkt statt
CREATOR	Name des Benutzers der dieses Objekt angelegt hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
BACKLOG_HANDLING	Der Backlog bestimmt wie sich das Time Scheduling System nach einer Serverdowntime verhält
SUSPEND_LIMIT	Die Zeit die definiert wann Jobs, z. B. nach einem Downtime, suspended submitted werden.
EFFECTIVE_SUSPEND_LIMIT	Die Zeit die definiert wann Jobs, z. B. nach einem Downtime, suspended submitted werden.
CALENDAR	Dieses Flag gibt an, ob Kalendereinträge erzeugt werden
CALENDAR_HORIZON	Die definierte Länge der Periode in Tagen für die ein Kalender erstellt wird
EFFECTIVE_CALENDAR_HOR	I 加 effektive Länge der Periode in Tagen für die ein Kalender erstellt wird
Fortsetzung auf der nächsten Se	eite

show scheduled event

User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CALENDAR_TABLE	Die Tabelle mit nächsten Startzeitpunkten

Tabelle 22.53.: Beschreibung der Output-Struktur des show scheduled event Statements

show scope

show scope

Zweck

Zweck

Das show scope Statement wird eingesetzt um detaillierte Informationen über einen Scope zu bekommen.

Syntax

Syntax

Die Syntax des show scope Statements ist

```
show < scope serverpath | job server serverpath > [ with EXPAND ]
```

```
EXPAND:
```

```
expand = none
| expand = < ( id {, id} ) | all >
```

Beschreibung

Beschreibung

Mit dem show scope Statement bekommt man ausführliche Informationen über den Scope.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OWNER	Die Gruppe die Eigentümer des Objektes ist
TYPE	Der Typ des Scopes
INHERIT_PRIVS	Vom übergeordneten Ordner zu erbende Privi-
	legien
IS_TERMINATE	Dieser Flag zeigt an ob ein Terminierungsauf-
	trag vorliegt
IS_SUSPENDED	Zeigt an ob der Scope suspendet ist
IS_ENABLED	Nur wenn der Enable Flag auf Yes gesetzt ist
	kann sich der Jobserver am Server anmelden

Fortsetzung auf der nächsten Seite

Feld	Beschreibung
IS_REGISTERED	Gibt an ob der Jobserver ein Register Befeh
	gesendet hat
IS_CONNECTED	Zeigt an ob der Jobserver Connected ist
HAS_ALTERED_CONFIG	Die Konfiguration weicht im Server von der ak
	tuellen im Jobserver ab
STATE	Hierbei handelt es sich um den aktuellen Statu
	der Resource in diesem Scope
PID	Bei dem PID handelt es sich um die Prozess
	Identifikationsnummer des Jobserverprozesse
	auf dem jeweiligen Hostsystem
NODE	Der Node gibt an auf welchem Rechner der Job
	server läuft. Dieses Feld hat einen rein doku
	mentativen Charakter
IDLE	Die Zeit die seit dem letzten Befehl verganger
TDD: 40.0	ist. Das gilt nur für Jobserver
ERRMSG	Hierbei handelt es sich um eine Fehlermeldung
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
CREATOR	Name des Benutzers der dieses Objekt angeleg hat
CREATE_TIME	Datum und Uhrzeit der Erstellung
CHANGER	Name des Benutzers der dieses Objekt zuletz geändert hat
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält
RESOURCES	Hier werden die Resourcen die in diesem Scope
	vorhanden sind angezeigt
	Siehe auch Tabelle 22.55 auf Seite 349
CONFIG	Im Tab Config steht die Konfiguration des Job
	servers beschrieben
	Siehe auch Tabelle 22.56 auf Seite 350
CONFIG_ENVMAPPING	In diesem Tab wir konfiguriert ob und unte
	welchem Namen die Umgebungsvariable
	sichtbar sind
	Siehe auch Tabelle 22.57 auf Seite 350
PARAMETERS	Im Tab Parameters können zusätzliche Informa
	tionen zu einer Resource gespeichert werden

User Commands show scope

Fortsetzung der vorherigen Seite		
Feld	Beschreibung	
	Siehe auch Tabelle 22.58 auf Seite 351	

Tabelle 22.54.: Beschreibung der Output-Struktur des show scope Statements

RESOURCES Das Layout der RESOURCES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NR_ID	Id der Named Resource
NAME	Name der Named Resource
USAGE	Hierbei handelt es sich um den Gebrauch der Named Resource (STATIC, SYSTEM oder SYN- CHRONIZING)
NR_PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf diese Named Resource enthält
TAG	Der Tag ist ein optionaler Kurzname für die Resource
OWNER	Die Gruppe die Eigentümer des Objektes ist
LINK_ID	Id der Ressource auf die verwiesen wird
LINK_SCOPE	Scope der Ressource auf die verwiesen wird
STATE	Der Resource State in dem sich die Resource befindet
REQUESTABLE_AMOUNT	Die Menge der Resourcen die maximal von einem Job angefordert werden darf
AMOUNT	Die aktuelle Menge die zur Verfügung steht
FREE_AMOUNT	Die freie Menge die allokiert werden darf
TOTAL_FREE_AMOUNT	Free amount available for allocations including free amount of pooled resources if it is a pool
IS_ONLINE	Hierbei handelt es sich um den Verfüg- barkeitsstatus der Resource
FACTOR	Dies ist der Korrekturfaktor mit dem ange- forderte Amounts multipliziert werden
TIMESTAMP	Der Timestamp gibt die Zeit des letzten Statuswechsels einer Resource an
SCOPE	Der Scope in dem die Resource angelegt ist

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
MANAGER_ID	Id des Managing Pools
MANAGER_NAME	Name des Managing Pools
MANAGER_SCOPENAME	Name des Scopes in dem der managing Pool angelegt wurde
HAS_CHILDREN	Flag das anzeigt ob ein Pool Child Resources/Pools managed. Wenn es kein Pool ist, ist es immer FALSE
POOL_CHILD	Dieses Flag zeigt an ob die gezeigte Resource ein Child vom Pool ist
TRACE_INTERVAL	Das trace Interval ist die minimale Zeit zwischen dem Scheiben von Trace Records in Sekunden
TRACE_BASE	Die trace Base ist es die Basis für den Auswertungszeitraum (B)
TRACE_BASE_MULTIPLIER	Der base Multiplier bestimmt den Multiplikationsfaktor (M) von der trace Base
TD0_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^0$ Sekunden
TD1_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^1$ Sekunden
TD2_AVG	Die durchschnittliche Resourcebelegung der letzten $B*M^2$ Sekunden
LW_AVG	Die durchschnittliche Resourcebelegung seit dem letzten Schreiben eines Trace Records
LAST_WRITE	Zeitpunkt des letzen Schreibens eines Trace Records
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.55.: Output-Struktur der show scope Subtabelle

CONFIG Das Layout der CONFIG Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
KEY	Der Name der Konfigurationsvariable
VALUE	Der Wert der Konfigurationsvariable
LOCAL	Zeigt an ob der Key Value Pairs local definiert
	ist oder übergeordnet
Fortsetzung auf der nächsten S	Seite

User Commands show scope

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ANCESTOR_SCOPE	Das ist der Scope in dem der Key Value Pairs
	definiert ist
ANCESTOR_VALUE	Das ist der Wert der übergeordnet definiert ist

Tabelle 22.56.: Output-Struktur der show scope Subtabelle

CONFIG_ENVMAPPING Das Layout der CONFIG_ENVMAPPING Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
KEY	Name der Umgebungsvariable
VALUE	Name der Umgebungsvariable die gesetzt werden soll
LOCAL	Zeigt an ob der Key Value Pairs local definiert ist oder übergeordnet
ANCESTOR_SCOPE	Das ist der Scope in dem der Key Value Pairs definiert ist
ANCESTOR_VALUE	Das ist der Wert der übergeordnet definiert ist

Tabelle 22.57.: Output-Struktur der show scope Subtabelle

PARAMETERS Das Layout der PARAMETERS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
EXPORT_NAME	Der Export Name definiert den Namen unter dem der Wert des Parameters in die Prozes- sumgebung exportiert wird.
ТҮРЕ	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
EXPRESSION	Expression für den Parameter des Typs EX- PRESSION
DEFAULT_VALUE	Der Default Value des Parameters
Fortsetzung auf der nächst	en Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
REFERENCE_TYPE	Typ des Objektes auf das referenziert wird
REFERENCE_PATH	Der Pfad des Objektes auf das referenziert wird
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes auf das referenziert wird
REFERENCE_PARAMETER	Name des Parameters auf den referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars
ID	Die Nummer des Repository Objektes
NAME	Name des Parameters
ТҮРЕ	Hierbei handelt es sich um die Art des Parameters
IS_LOCAL	True für lokale Parameter die nur für den Job selbst sichtbar sind
REFERENCE_TYPE	Typ des Objektes welches den Parameter referenziert
REFERENCE_PATH	Der Pfad des Objektes welches den Parameter referenziert
REFERENCE_PRIVS	Die Privilegien des Benutzers auf das Objektes welches den Parameter referenziert
REFERENCE_PARAMETER	Name des Parameters auf dem referenziert wird
COMMENT	Kommentar zum Objekt, wenn vorhanden
COMMENTTYPE	Typ des Kommentars

Tabelle 22.58.: Output-Struktur der show scope Subtabelle

show session

show session

Zweck

Zweck

Das show session Statement wird eingesetzt um mehr detaillierte Informationen über die spezifizierte oder die akutelle Session zu bekommen.

Syntax

Syntax

Die Syntax des show session Statements ist

show session [*sid*]

Beschreibung

Beschreibung

Mit dem show session Statement bekommt man ausführliche Informationen über die spezifizierte oder aktuelle Session.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
THIS	Bei This wird die eigene Session mit einem Stern
	angezeigt
SESSIONID	Hierbei handelt es sich um die Id der Session
START	Der Zeitpunkt ab dem die Session gestartet
	wurde
USER	Name des Users mit dem die Session
	angemeldet wurde
UID	Die Id des Users
IP	Die IP-Adresse des Rechners von dem die
	Verbindung gemacht wird
TXID	Die Id der Letzten oder aktuellen Transaktion
IDLE	Die Zeit in Sekunden die seit der letzten
	Transaktion vergangen ist
TIMEOUT	Die Zeit in Sekunden nach der eine automatis-
	che Abmeldung erfolgt. (0 bedeutet unlimitiert)

Fortsetzung auf der nächsten Seite

show session User Commands

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
STATEMENT	Das Statement das zur Zeit ausgeführt wird

Tabelle 22.59.: Beschreibung der Output-Struktur des show session Statements

show system

show system

Zweck

Zweck

Das show system Statement wird eingesetzt um Informationen über die aktuelle Konfiguration des laufenden Servers zu bekommen.

Syntax

Syntax

Die Syntax des show system Statements ist

show system

show system with lock

Beschreibung

Beschreibung

Mit dem show system Statement bekommt man ausführliche Informationen über die aktuelle Konfiguration des laufenden Servers.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
VERSION	Die aktuelle Version der Software
MAX_LEVEL	Die maximale Kompatibilitätsstufe der Soft-
	ware
NUM_CPU	Die Anzahl Prozessoren die in dem System
	vorhanden sind
MEM_USED	Die Menge benutzter Hauptspeicher
MEM_FREE	Die Menge freier Speicher
MEM_MAX	Die maximale Speichermenge die der Server in
	Anspruch nehmen kann
STARTTIME	Der Zeitpunkt in dem der Server gestartet
	wurde
UPTIME	Der Zeitpunkt ab dem der Server schon läuft
HITRATE	Der Hitrate im Environment Cache des Schedul-
	ing Threads
Fortsetzung auf der nächsten S	Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
LOCK_HWM	Der <i>lock_hwm</i> zeigt die High Water Mark der Anzahl aktive Sperren im System Dieses Feld ist nur dann relevant, wenn mehrere Writer Threads aktiv sind.
LOCKS_REQUESTED	Das feld <i>locks_requested</i> zeigt die Anzahl beantragte Sperren seit Server Startup. Dieses Feld ist nur relevant, wenn mehrere Writer Threads aktiv sind.
LOCKS_USED	Dieses Feld zeigt die Anzahl derzeit benutzte Sperren. Es ist nur relevant, wenn mehrere Writ- er Threads aktiv sind.
LOCKS_DISCARDED	Das Feld <i>locks_discarded</i> zeigt die Anzahl Sperren die freigegeben wurden, ohne sie für spätere Wiederbenutzung auf zu heben.
CNT_RW_TX	Die Anzahl R/W Transaktionen seit Server Startup.
CNT_DL	Die Anzahl Deadlocks seit Server Startup.
CNT_WL	Die Anzahl single threaded Write Worker Tansaktionen seit Server Startup.
WORKER	Eine Tabelle mit einer Liste der Worker Threads Siehe auch Tabelle 22.61 auf Seite 355
LOCKING STATUS	Dieses Feld wurde noch nicht beschrieben

Tabelle 22.60.: Beschreibung der Output-Struktur des show system Statements

WORKER Das Layout der WORKER Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
TYPE	Dieses Feld wurde noch nicht beschrieben
NAME	Der Name des Objektes
STATE	Der Status des Workers
TIME	Der Zeitpunkt ab dem der Worker in einem Sta-
	tus ist

Tabelle 22.61.: Output-Struktur der show system Subtabelle

show trigger

show trigger

Zweck

Zweck

Das show trigger Statement wird eingesetzt um detaillierte Informationen über den spezifizierten Trigger zu bekommen.

Syntax

Syntax

Die Syntax des show trigger Statements ist

```
show trigger triggername on TRIGGEROBJECT [ < noinverse | inverse > ]
```

TRIGGEROBJECT:

```
resource resourcepath in folderpath job definition folderpath object monitor objecttypename resource resourcepath in serverpath
```

Beschreibung

Beschreibung

Mit dem show trigger Statement bekommt man ausführliche Informationen über den spezifizierten Trigger.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
OBJECTTYPE	Der Typ des Objektes in dem der Trigger definiert ist
OBJECTNAME	Kompletter Pfadname des Objektes in dem der Trigger definiert ist
ACTIVE	Gibt an ob der Trigger aktiv ist, oder nicht
ACTION	Als Trigger Aktion erfolgt entweder der Submit
	eines Jobs, oder ein Rerun des triggernden Jobs
Fortsetzung auf der nächsten S	Seite

Fortsetzung der vorherigen Seite				
Feld	Beschreibung			
SUBMIT_TYPE	Der Objekttyp der submitted wird, wenn getriggert wird			
SUBMIT_NAME	Der Objekttyp der submitted wird, wenn getriggert wird			
SUBMIT_SE_OWNER	Der Besitzer des Objektes der submitted wird			
SUBMIT_PRIVS	Die Privilegien auf das zu submittende Objekt			
MAIN_TYPE	Typ des main Jobs (Job/Batch)			
MAIN_NAME	Name des main Jobs			
MAIN_SE_OWNER	Owner des main Jobs			
MAIN_PRIVS	Privilegien auf den main Job			
PARENT_TYPE	Typ des parent Jobs (Job/Batch)			
PARENT_NAME	Name des parent Jobs			
PARENT_SE_OWNER	Owner des parent Jobs			
PARENT_PRIVS	Privilegien auf den parent Job			
TRIGGER_TYPE	Der Trigger Typ der beschreibt wann gefeuert wird			
MASTER	Zeigt an ob der Trigger einen Master oder ein Child submitted			
IS_INVERSE	Im Falle eines inverse Triggers gehört der Trigger dem getriggerten Job. Der Trigger kann so als Art Callback Funktion gesehen werden. Der Flag hat keinen Einfluß auf die Funktion des Triggers.			
SUBMIT_OWNER	Die Eigentümergruppe die bem Submitted Entity eingesetzt werden muss			
IS_CREATE	Zeigt an ob der Trigger auf create Events reagiert			
IS_CHANGE	Zeigt an ob der Trigger auf change Events reagiert			
IS_DELETE	Zeigt an ob der Trigger auf delete Events reagiert			
IS_GROUP	Zeigt an ob der Trigger die Events als Gruppe behandelt			
MAX_RETRY	Die maximale Anzahl von Trigger Auslösungen in einem einzelnen Submitted Entity			
SUSPEND	Spezifiziert ob das submittete Objekt suspended wird			

show trigger

Fortsetzung der vorherige Feld	Beschreibung		
RESUME_AT	Zeitpunkt des automatischen Resume		
RESUME_IN	Anzahl Zeiteinheite bis zum automatischen Resume		
RESUME_BASE	Zeiteinheit zu resume_in		
WARN	Spezifiziert ob eine Warnung ausgegeben werden muss wenn das Feuerlimit erreicht ist		
LIMIT_STATE	Spezifiziert den Status der vom auslösender Jobs angenommen wird, wenn das Fire Limiterreicht wird. Hat der Job bereit einen finaler Status, wird diese Einstellung ignoriert. Steht der Wert auf NONE , wird keine Statusänderung vorgenommen.		
CONDITION	Konditionaler Ausdruck um die Trigger Kondi tion zu definieren		
CHECK_AMOUNT	Die Menge der CHECK_Base Einheiten undie Kondition bei nicht Synchronen Trigger zu überprüfen		
CHECK_BASE	Einheiten für den CHECK_AMOUNT		
COMMENT	Kommentar zum Objekt, wenn vorhanden		
COMMENTTYPE	Typ des Kommentars		
CREATOR	Name des Benutzers der dieses Objekt angeleg hat		
CREATE_TIME	Datum und Uhrzeit der Erstellung		
CHANGER	Name des Benutzers der dieses Objekt zuletz geändert hat		
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung		
STATES	Tabelle von States die zum Auslösen des Trig gers führen		
	Siehe auch Tabelle 22.63 auf Seite 359		

Tabelle 22.62.: Beschreibung der Output-Struktur des show trigger Statements

STATES Das Layout der STATES Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung	
ID	Die Nummer des Repository Objektes	
FROM_STATE	Der Trigger feuert wenn das der alte Resource	
	State ist	
Fortsetzung auf der näch	sten Seite	

show trigger User Commands

Fortsetzung der vorherigen	n Seite
Feld Beschreibung	
TO_STATE	Der Trigger feuert wenn das der neue Resource State oder der Exit State des Objektes ist

Tabelle 22.63.: Output-Struktur der show trigger Subtabelle

show user

show user

Zweck

Zweck

Das show user Statement wird eingesetzt um detaillierte Informationen über den Benutzer anzuzeigen.

Syntax

Syntax

Die Syntax des show user Statements ist

show user [*username*]

Beschreibung

Beschreibung

Mit dem show user Statement bekommt man ausführliche Informationen über den Benutzer.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung			
ID	Die Nummer des Repository Objektes			
NAME	Der Name des Objektes			
IS_ENABLED	Flag das anzeigt ob es dem Benutzer erlaubt ist sich anzumelden			
DEFAULT_GROUP	Die Default Gruppe der Benutzer die die Eigentümer des Objektes benutzen			
CREATOR	Name des Benutzers der dieses Objekt angelegt hat			
CREATE_TIME	Datum und Uhrzeit der Erstellung			
CHANGER	Name des Benutzers der dieses Objekt zuletzt geändert hat			
CHANGE_TIME	Datum und Uhrzeit der letzten Änderung			
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält			

360 I show user

show user User Commands

Fortsetzung der vorherigen Seite				
Feld	Beschreibung			
MANAGE_PRIVS	Tabelle der manage Privilegien			
	Siehe auch Tabelle 22.65 auf Seite 361			
GROUPS	Tabelle der Gruppen zu denen der Benutzer			
	gehört			
	Siehe auch Tabelle 22.66 auf Seite 361			
COMMENTTYPE	Typ des Kommentars			
COMMENT	Kommentar zum Objekt, wenn vorhanden			
	Siehe auch Tabelle 22.67 auf Seite 361			

Tabelle 22.64.: Beschreibung der Output-Struktur des show user Statements

MANAGE_PRIVS Das Layout der MANAGE_PRIVS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte
	auf dieses Objekt enthält

Tabelle 22.65.: Output-Struktur der show user Subtabelle

GROUPS Das Layout der GROUPS Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung
ID	Die Nummer des Repository Objektes
NAME	Der Name des Objektes
PRIVS	Zeichenkette die Kürzel für die Benutzerrechte auf dieses Objekt enthält

Tabelle 22.66.: Output-Struktur der show user Subtabelle

COMMENT Das Layout der COMMENT Tabelle wird in untenstehender Tabelle gezeigt.

Feld	Beschreibung		
TAG	Dieses Feld wurde noch nicht beschrieben		
COMMENT	Kommentar zum Objekt, wenn vorhanden		

Tabelle 22.67.: Output-Struktur der show user Subtabelle

23. shutdown commands

shutdown

shutdown

Zweck

Zweck

Das shutdown Statement wird eingesetzt um den adressierten Jobserver anzuweisen sich zu beenden.

Syntax

Syntax

Die Syntax des shutdown Statements ist

shutdown serverpath

Beschreibung

Beschreibung

Mit dem shutdown Statement beendet man den adressierten Jobserver.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

24. stop commands

stop server

stop server

Zweck

Zweck

Das stop server Statement wird eingesetzt um den Server anzuweisen sich zu beenden.

Syntax

Syntax

Die Syntax des stop server Statements ist

stop server

stop server kill

Beschreibung

Beschreibung

Mit dem stop server Statement beendet man den Server. Sollte dies aus irgendeinem Grund nicht richtig funktionieren, kann der Server auch hart beended werden durch **kill** zu spezifizieren.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

25. submit commands

submit

submit

Zweck

Zweck

Das submit Statement wird eingesetzt um einen Master Batch oder Job, sowie alle definierten Children, auszuführen.

Syntax

Syntax

Die Syntax des submit Statements ist

Beschreibung

Beschreibung

Das submit Statement wird benutzt um einen Job oder Batch zu submitten. Es existieren zwei Formen des Submit-Kommandos.

- Die erste Form wird von Benutzern, welche auch Programme sein können und dem Time Scheduling Module genutzt. Diese Form submitted Master Jobs und Batches.
- Die zweite Form des Statements wird von Jobs genutzt, um dynamische Children zu submitten.

368 I submit

check only Die check only Option wird benutzt um zu überprüfen ob ein Master Submittable Batch oder Job submitted werden kann. Das bedeutet, es wird geprüft ob alle Abhängigkeiten erfüllt werden können und alle referenzierten Parameter definiert sind.

Es wird nicht überprüft ob die Jobs in irgendeinem Scope ausgeführt werden können oder nicht. Das ist eine Situation die jederzeit zur Laufzeit auftreten kann.

Eine positive Rückmeldung bedeutet, dass der Job oder Batch aus Sicht des Systems submitted werden kann.

Die check only Option kann nicht in einem Job Kontext benutzt werden.

childtag Die childtag Option wird von Jobs benutzt, um verschiedene Instanzen von dem selben Scheduling Entity zu submitten und um zwischen ihnen unterscheiden zu können.

Es führt zu einem Fehler, wenn der gleiche Scheduling Entity doppelt submitted wird, wenn sich der Childtag nicht unterscheidet. Der Inhalt des Childtags hat keine weitere Bedeutung für den Scheduling System.

Die maximale länge eines Childtags beträgt 70 Zeichen. Die Childtag Option wird im Falle eines Master Submits ignoriert.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

nicevalue Die nicevalue Option definiert eine Korrektur die für die Berechnung der Prioritäten des Jobs und seinen Children benutzt wird. Es sind Werte von -100 bis 100 erlaubt.

parameter Die parameter Option wird benutzt um den Wert von Job Parameter beim Submit zu spezifizieren. Die Parameter werden in dem Scope des Master Batches oder Jobs gesetzt. Das Bedeutet, wenn Parameter, die nicht in dem Master Batch oder Job definiert sind, spezifiziert werden, sind diese Parameter unsichtbar für Children.

submittag Wenn der submittag spezifiziert ist, muss er eine eindeutige Bezeichnung für den Submitted Entity haben. Dieser Tag wurde, um imstande zu sein Jobs und Batches programmatisch zu submitten und um den Job oder Batch, mit demselben Tag, nach einem Absturz von einem der Komponenten neu zu submitten. Wenn die Submission des Jobs das erste mal erfolgreich war, wird der zweite Submit einen Fehler melden. Wenn nicht, wird der zweite Submit erfolgreich sein.

submit | 369

submit

unresolved Die unresolved Option spezifiziert wie der Server bei nicht auflösbaren Abhängigkeiten reagieren sollte. Diese Option wird hauptsächlich benutzt, wenn Teile eines Batches nach Reparaturarbeiten submitted werden. Der Fehlerhafte Teil wird normal gecancelt und dann als Master Run neu submittet. Die vorherigen Abhängigkeiten müssen in diesem Fall ignoriert werden, andernfalls wird der Submit scheitern.

suspend Die suspend Option wird benutzt um Jobs oder Batches zu submitten und sie zur selben Zeit zu suspenden. Wenn nichts festgelegt wurde, wird nicht suspended. Dies kann explizit zur Submit-Zeit spezifiziert werden. Wenn ein Job oder Batch suspended wurde wird er, sowie auch seine Children, nicht gestartet. Wenn ein Job bereits läuft, wird er keinen Final State erreichen, wenn er suspended ist.

resume Die resume Option kann zusammen mit der suspend Option verwendet werden um eine verzögerte Ausführung zu bewirken. Es gibt dabei zwei Möglichkeiten. Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll. Mit dieser Option kann die at-Funktionalität ohne das Anlegen eines Schedules nachgebildet werden.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Id des Submitted Entities

Tabelle 25.1.: Beschreibung der Output-Struktur des submit Statements

370 I submit

26. suspend commands

suspend

suspend

Zweck

Zweck

Das suspend Statement wird eingesetzt um zu verhindern das weitere Jobs von diesem Jobserver ausgeführt werden. Siehe das resume Statement auf Seite 270.

Syntax

Syntax

Die Syntax des suspend Statements ist

suspend serverpath

Beschreibung

Beschreibung

Mit dem suspend Statement wird verhindert dass weitere Jobs von diesem Jobserver ausgeführt werden.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Teil III. Jobserver Commands

27. Jobserver Commands

Jobserver Commands

alter job

alter job

Zweck

Zweck

Das alter job Statement wird benutzt um Eigenschaften des spezifizierten Jobs zu ändern. Es wird von den Job Administratoren, Jobservern und von dem Job selbst benutzt.

Syntax

Syntax Die Syntax des alter job Statements ist

```
alter job jobid
with WITHITEM {, WITHITEM}
alter job
with WITHITEM {, WITHITEM}
WITHITEM:
     < suspend | suspend restrict | suspend local | suspend local restrict >
     cancel
     clear warning
     comment = string
     error text = string
     exec pid = pid
     exit code = signed_integer
     exit state = statename [ force ]
     ext pid = pid
     ignore resource = ( id {, id} )
     ignore dependency = ( jobid [ recursive ] {, jobid [ recursive ]} )
     kill
     nicevalue = signed_integer
     priority = integer
     renice = signed_integer
     rerun [recursive]
     resume
     < noresume | resume in period | resume at datetime >
     run = integer
     state = JOBSTATE
     timestamp = string
     warning = string
```

JOBSTATE:

alter job

Jobserver Commands

broken active
broken finished
dependency wait
error
finished
resource wait
running
started
starting
synchronize wait

Beschreibung

Das alter job Kommando wird für mehrere Zwecke genutzt. Als erstes verwenden Jobserver dieses Kommando um den Ablauf eines Jobs zu dokumentieren. Alle Statuswechsel eines Jobs während der Zeit in der der Job innerhalb der Zuständigkeit eines Jobservers fällt, werden mittels des alter job Kommandos ausgeführt. Zweitens werden einige Änderungen wie z. B. das Ignorieren von Abhängigkeiten oder Resourcen sowie das Ändern der Priorität eines Jobs manuell von einem Administrator ausgeführt.

Der Exit State eines Jobs in einem Pending State kann von dem Job selbst gesetzt werden, bzw. von einem Prozess welcher die Job ID und den Key des zu ändernden Jobs kennt.

cancel Die cancel Option wird benutzt um den adressierten Job und alle nicht Final Children zu canceln. Ein Job kann nur gecancelled werden wenn weder der Job selbst noch einer seiner Kinder aktiv ist.

Wenn ein Scheduling Entity von dem gecancelten Job abhängig ist, kann er unreachable werden. In diesem Fall erhält der abhängige Job nicht den im Exit State Profile definierten Unreachable Exit State, sondern wird in den Job Status "Unreachable" versetzt. Es ist Aufgabe des Operators diese Jobs nun mittels des Ignorierens von Abhängigkeiten wieder in den Job Status "Dependency Wait" zu versetzen, oder aber diese Jobs auch zu canceln.

Gecancelte Jobs werden wie Final Jobs ohne Exit State betrachtet. Das bedeutet, die Parents eines gecancelten Jobs werden final, ohne den Exit State des gecancelten Jobs zu berücksichtigen. Die abhängigen Jobs der Parents laufen in diesem Fall normal weiter.

Die cancel Option kann nur von Benutzern genutzt werden.

comment Die comment Option wird benutzt um eine Aktion zu dokumentieren oder um dem Job einen Comment zuzufügen. Comments können maximal 1024 Zeichen lang sein. Es kann eine beliebige Anzahl Comments für einen Job gespeichert werden.

Beschreibung

Einige Comments werden automatisch gespeichert. Wenn z. B. ein Job einen Restartable State erreicht, wird ein Protokoll geschrieben, um diesen Fakt zu dokumentieren.

error text Die error text Option wird benutzt um Fehlerinformation zu einem Job zu schreiben. Dieses kann von dem verantwortlichen Jobserver oder einem Benutzer gemacht werden. Der Server kann diesen Text auch selbst schreiben. Diese Option wird normalerweise benutzt, wenn der Jobserver den entsprechenden Prozess nicht Starten kann. Mögliche Fälle sind die Unmöglichkeit zum definierten Working Directory zu wechseln, die Unauffindbarkeit des ausführbaren Programmes oder Fehler beim Öffnen des Error Logfiles.

exec pid Die exec pid Option wird ausschließlich von dem Jobserver benutzt um die Prozess ID des Kontroll Prozesses innerhalb des Servers zu setzen.

exit code Die exit code Option wird von dem Jobserver benutzt um dem Repository Server mitzuteilen mit welchem Exit Code sich ein Prozess beendet hat. Der Repository Server berechnet jetzt den zugehörigen Exit State aus dem verwendeten Exit State Mapping.

exit state Die Exit State Option wird von Jobs in einem Pending State benutzt, um ihren State auf einen anderen Wert zu setzen. Dies wird normalerweise ein Restartable oder Final State sein.

Alternativ dazu kann diese Option von Administratoren benutzt werden, um den State von einem nonfinal Job zu setzen.

Sofern das Force Flag nicht benutzt wird, sind die einzigen States die gesetzt werden können, die States welche, durch die Anwendung des Exit State Mappings auf irgendeinem Exit Code, theoretisch erreichbar sind. Der gesetzte State muss im Exit State Profile vorhanden sein.

ext pid Die ext pid Option wird ausschließlich von dem Jobserver genutzt, um die Prozess ID des gestarteten Benutzerprozesses zu setzen.

ignore resource Die ignore resource Option wird benutzt um einzelne Resource Requests aufzuheben. Die ignorierte Resource wird nicht mehr beantragt.

Wenn Parameter einer Resource referenziert werden, kann diese Resource nicht ignoriert werden.

Wenn ungültige ID's spezifiziert wurden, wird dies übergangen. Alle anderen spezifizierten Resources werden ignoriert. Ungültige ID's in diesem Kontext sind ID's von Resources die von dem Job nicht beantragt werden.

Das Ignorieren von Resources wird protokolliert.

ignore dependency Die ignore dependency Option wird benutzt um definierte Dependencies zu ignorieren. Wenn das **recursive** flag benutzt wird, ignorieren nicht nur der Job oder Batch selbst, sondern auch seine Kinder die Dependencies.

kill Die kill Option wird benutzt um den definierten Kill Job zu submitten. Wenn kein Kill Job definiert ist, ist es nicht möglich den Job vom BICsuite aus erzwungenermaßen zu terminieren. Natürlich muss der Job aktiv sein, das bedeutet, der Jobstate muss **running**, **killed** oder **broken_active** sein. Die letzten beiden States sind keine regulären Fälle.

Wenn ein Kill Job submitted wurde, ist der Jobstate to_kill. Nachdem der Kill Job beendet wurde, wird der Jobstate des killed Jobs in den State killed gesetzt, es sei denn er ist beendet, dann wird der Jobstate finished oder final sein. Das bedeutet, der Job mit dem Jobstate killed bedeutet, dass er immer noch running ist und dass mindestens ein Versuch gemacht wurde, den Job zu terminieren.

nicevalue Die nicevalue Option wird benutzt um die Priorität oder den Nicevalue eines Jobs oder Batches und allen ihren Children zu ändern. Hat ein Child mehrere Parents, kann eine Änderung, muss aber nicht, in dem Nicevalue von einem der Parents Auswirkungen auf die Priorität des Childs haben. In dem Fall das es mehrere Parents gibt wird das maximale Nicevalue gesucht.

Also, wenn Job C drei Parents P1, P2 und P3 hat und P1 setzt einen Nicevalue von 0, P2 einen von 10 und P3 einen von -10, ist der effektive Nicevalue -10. (Umso niederiger der Nicevalue, umso besser). Wenn der Nicevalue von P2 auf -5 geändert wird, passiert nichts, weil die -10 von P3 besser als -5 ist. Wenn jetzt der Nicevalue von P3 auf 0 sinkt, wird die neue effektive Nicevalue für Job C -5.

Die Nicevalues können Werte zwischen -100 und 100 haben. Werte die diese Spanne übersteigen, werden stillschweigend angepasst.

priority Die priority Option wird benutzt, um die (statische) Priorität eines Jobs zu ändern. Weil Batches und Milestones nicht ausgeführt werden, haben Prioritäten keine Bedeutung für sie.

Ein Wechsel der Priorität betrifft nur den geänderten Job. Gültige Werte liegen zwischen 0 und 100. Dabei korrespondiert 100 mit der niedrigsten Priorität und 0 mit der höchsten Priorität.

Bei der Berechnung der dynamischen Priorität eines Jobs, startet der Scheduler mit der statischen Priorität und passt dies, entsprechend der Zeit in der der Job schon wartet, an. Wenn mehr als ein Job die gleiche dynamische Priorität hat, wird der Job mit der niedrigsten Job ID als erster gescheduled.

renice Die renice Option gleicht der nicevalue Option mit dem Unterschied, das die renice Option relativ arbeitet, während die nicevalue Option absolut arbeitet. Wenn einige Batches einen Nicevalue von 10 haben bewirkt eine renice von -5,

alter job

dass die Nicevalue auf 5 zunimmt. (Zunahme, weil je niedriger die Nummer, desto höher die Piorität)

rerun Die rerun Option wird benutzt um einen Job in einem Restartable State neu zu starten. Der Versuch einen Job, der nicht restartable ist, neu zu starten, führt zu einer Fehlermeldung. Ein Job ist restartable, wenn er in einem Restartable State oder in einem **error** oder **broken_finished** Jobstate ist.

Wenn das **recursive** flag spezifiziert ist, wird der Job selbst und alle direkten und indirekten Children, die in einem Restartable State sind, neu gestartet. Wenn der Job selbst Final ist, wird das in dem Fall nicht als Fehler betrachtet. Es ist also möglich Batches rekursiv neu zu starten.

resume Die resume Option wird benutzt um einen Suspended Job oder Batch zu reaktivieren. Es gibt dabei zwei Möglichkeiten. Erstens kann der suspended Job oder Batch sofort reaktiviert werden, und zweitens kann eine Verzögerung eingestellt werden.

Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll.

Für die Spezifikation einer Zeit siehe auch die Übersicht auf Seite 6.

Die resume Option kann zusammen mit der suspend Option verwendet werden. Dabei wird der Job suspended und nach der (bzw. zur) spezifizierten Zeit wieder resumed.

run Die run Option wird von dem Jobserver benutzt zwecks der Sicherstellung, dass der geänderte Job mit der aktuellen Version übereinstimmt.

Theoretisch ist es möglich, dass nachdem ein Job von einem Jobserver gestartet wurde, der Computer abstürzt. Um die Arbeit zu erledigen wird der Job mittels eines manuellen Eingriffs, von einem anderen Jobserver, neu gestartet. Nach dem hochfahren des ersten Systems, kann der Jobserver versuchen den Jobstate nach broken_finished zu ändern, ohne über das Geschehen nach dem Absturz bescheid zu wissen. Das Benutzen der run Option, verhindert nun das fälschlich Setzen des Status.

state Die state Option wird hauptsächlich von Jobservern benutzt, sie kann aber auch von Administratoren benutzt werden. Es wird nicht empfohlen das so zu machen, es sei denn Sie wissen genau was Sie tun.

Die übliche Prozedur ist, das der Jobserver den State eines Jobs von **starting** nach **started**, von **started** nach **running** und von **running** nach **finished** setzt. Im Falle eines Absturzes oder anderen Problemen ist es möglich dass der Jobserver einen Job in einen **broken_active** oder **broken_finished** State setzt. Das bedeutet, der Exit Code von dem Prozess steht nicht zur Verfügung und der Exit State muss manuell gesetzt werden.

suspend Die suspend Option wird benutzt um einen Batch oder Job zu suspendieren. Sie arbeitet immer rekursiv. Wenn ein Parent Suspended ist, sind auch alle Children suspended. Die resume Option wird benutzt um die Situation zu umzukehren.

Die **restrict** Option bewirkt, dass nur Benutzer der ADMIN Gruppe die Suspendierung wieder aufheben können.

timestamp Die timestamp Option wird von dem Jobserver benutzt um die Timestamps der Statuswechsel zu setzen, gemäß der lokalen Zeit aus Sicht des Jobservers.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Jobserver Commands

alter jobserver

alter jobserver

Zweck

Zweck

Das alter jobserver Statement wird eingesetzt um die Eigenschaften eines Jobservers zu ändern.

Syntax

Syntax

Die Syntax des alter jobserver Statements ist

```
alter [ existing ] job server
with < fatal | nonfatal > error text = string

alter [ existing ] job server
with dynamic PARAMETERS

PARAMETERS:
    parameter = none
    | parameter = ( PARAMETERSPEC {, PARAMETERSPEC} )

PARAMETERSPEC:
    parametername = < string | number >
```

Beschreibung

Beschreibung

Das alter scope Kommando ist ein Benutzerkommando sowie ein Jobserverkommando. Es wird als Benutzerkommando benutzt um die Konfiguration oder andere Eigenschaften eines Scopes oder Jobservers zu ändern. Weitere Details sind im create scope Kommando auf Seite 135 beschrieben. Die Syntax von dem Benutzerkommando entspricht der ersten Form des alter scope Kommandos.

Als Jobserver Kommando wird es benutzt um den Server über Fehler zu benachrichtigen. Wird der Fatal Flag benutzt, bedeutet das, dass sich der Jobserver beendet. In dem anderen Fall läuft der Jobserver weiter.

Die dritten Form des alter jobserver Kommandos wird auch vom Jobserver benutzt. Der Jobserver veröffentlicht die Werte seines dynamischen Parameters. Der Server verwendet veröffentlichte Werte um Parameter in der Kommandozeile und Logfile-Angaben beim Abholen eines Jobs aufzulösen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

connect

Jobserver Commands

connect

Zweck

Das connect Statement wird eingesetzt um einen Jobserver bei dem Server zu Zweck authentifizieren.

Syntax

Die Syntax des connect Statements ist

Syntax

```
connect job server serverpath . servername identified by string [ with
WITHITEM {, WITHITEM} ]

WITHITEM:
    command = ( sdms-command )
    | protocol = PROTOCOL
    | session = string
    | timeout = integer
    | < trace | notrace >
    | trace level = integer

PROTOCOL:
    json
    | line
    | perl
    | python
```

Beschreibung

serial xml

Das connect Kommando wird benutzt um den verbundenen Prozess beim Server zu authentifizieren. Es kann wahlweise ein Kommunikationsprotokoll spezifiziert werden. Das default protocol ist **line**.

Beschreibung

Das ausgewählte Protokoll definiert die Form des Outputs. Alle Protokolle, außer serial, liefern ASCII Output. Das Protokoll serial liefert einen serialized Java Objekt zurück.

Beim Connect kann gleich ein auszuführendes Kommando mitgegeben werden. Als Output des Connect Kommandos wird in dem Fall der Output des mitgegebenen Kommandos genutzt. Falls das Kommando fehl schlägt, der Connect aber gültig war, bleibt die Connection bestehen.

Im folgenden ist für alle Protokolle, außer für das **serial** Protokoll, ein Beispiel gegeben.

connect | 383

Jobserver Commands

connect

Das line protocol Das line protocol liefert nur einen ASCII Text als Ergebnis von einem Kommando zurück.

```
connect donald identified by 'duck' with protocol = line;
Connect
CONNECT_TIME : 19 Jan 2005 11:12:43 GMT
Connected
SDMS>
```

Das XML protocol Das XML protocol liefert eine XML-Struktur als Ergebnis eines Kommandos zurück.

```
connect donald identified by 'duck' with protocol = xml;
<OUTPUT>
<DATA>
<TITLE>Connect</TITLE>
<RECORD>
<CONNECT_TIME>19 Jan 2005 11:15:16 GMT</CONNECT_TIME></RECORD>
</DATA>
<FEEDBACK>Connected</FEEDBACK>
</OUTPUT>
```

Das python protocol Das python protocol liefert eine python-Struktur, welche durch die python eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = python;
{
'DATA':
{
'TITLE': 'Connect',
'DESC': [
'CONNECT_TIME'
],
'RECORD': {
'CONNECT_TIME': '19 Jan 2005 11:16:08 GMT'}
}
, 'FEEDBACK': 'Connected'
}
```

384 I connect

Jobserver Commands

connect

Das perl protocol Das perl protocol liefert eine perl Struktur, welche durch die perl eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = perl;
{
'DATA' => {
   'TITLE' => 'Connect',
   'DESC' => [
   'CONNECT_TIME' |
],
   'RECORD' => {
   'CONNECT_TIME' => '19 Jan 2005 11:19:19 GMT'}
}, 'FEEDBACK' => 'Connected'
}
```

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

connect I 385

deregister

deregister

Zweck

Zweck

Das deregister Statement wird eingesetzt um den Server zu benachrichtigen, das der Jobserver keine Jobs mehr ausführt. Siehe das register Statement auf Seite 248.

Syntax

Syntax

Die Syntax des deregister Statements ist

deregister serverpath . servername

Beschreibung

Beschreibung

Das deregister Statement wird genutzt um den Server über einen, mehr oder weniger, permanenten Ausfall eines Jobservers zu informieren.

Diese Nachricht hat verschiedene Serveraktionen zur Folge. Als erste werden alle running Jobs des Jobservers, d.h. Jobs im Status **started**, **running**, **to_kill** und **killed**, auf den Status **broken_finished** gesetzt. Jobs im Status **starting** werden wieder auf **runnable** gesetzt. Dann wird der Jobserver aus der Liste der Jobservers die Jobs verarbeiten können entfernt, so dass dieser Jobserver im Folgenden auch keine Jobs mehr zugeteilt bekommt. Als Nebeneffekt werden Jobs, die aufgrund Resourceanforderungen nur auf diesen Jobserver laufen können, in den Status **error**, mit der Meldung "Cannot run in any scope because of resource shortage", versetzt. Als letzte wird einen kompletten Reschedule ausgeführt um eine Neuverteilung von Jobs auf Jobservers herbei zu führen.

Durch erneutem Registrieren (siehe das register Statement auf Seite 248), wird der Jobserver erneut in die Liste der Jobs-verarbeitenden Jobservers eingetragen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

disconnect

Jobserver Commands

disconnect

Zweck

Das disconnect Statement wird eingesetzt um die Serververbindung zu beenden. Zweck

Syntax

Die Syntax des disconnect Statements ist

Syntax

disconnect

Beschreibung

Mit dem disconnect Statement kann die Verbindung zum Server beendet wer- Beschreibung den.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

Jobserver Commands

get next job

get next job

Zweck

Zweck

Die Zweck des get next job Statements ist, das nächste Kommando von dem Server zu holen.

Syntax

Syntax

Die Syntax des get next job Statements ist

get next job

Beschreibung

Beschreibung

Mit dem get next job Statement holt der Jobserver das nächste auszuführende Kommando vom Server.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Tabelle.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
COMMAND	Das vom Jobserver aus zu führende Kommando. (NOP, ALTER, SHUTDOWN, STARTJOB)
CONFIG	Geänderte Konfiguration. Dieser Wert ist nur im Falle eines ALTER Kommandos vorhanden
ID	Die Id des zu startenden Jobs; nur vorhanden beim Kommando STARTJOB
DIR	Das Working Directory des zu startenden Jobs; nur vorhanden beim Kommando STARTJOB
LOG	Das Logfile des zu startenden Jobs; nur vorhanden beim Kommando STARTJOB
LOGAPP	Indikator ob das Logfile mit Append geöffnet werden soll; nur vorhanden beim Kommando STARTJOB

Fortsetzung auf der nächsten Seite

Fortsetzung der vorherigen Seite	
Feld	Beschreibung
ERR	Das Errorlogfile des zu startenden Jobs; nur vorhanden beim Kommando STARTJOB
ERRAPP	Indikator ob das Errorlogfile mit Append geöffnet werden soll; nur vorhanden beim Kom- mando STARTJOB
CMD	Filename des zu startenden Executables; nur vorhanden beim Kommando STARTJOB
ARGS	Die Commandline Parameter des zu startenden Executables; nur vorhanden beim Kommando STARTJOB
ENV	Zusätzliche Eintrage für das Environment des zu startenden Executables; nur vorhanden beim Kommando STARTJOB
RUN	Nummer des Runs. Siehe auch Alter Job Statement auf seite 47; nur vorhanden beim Kommando STARTJOB
JOBENV	Liste von key value Paaren welche definiert, welche in der Job Definition definierten Umgebungsvariablen vor der Jobausführung gesetzt werden sollen.

Tabelle 27.1.: Beschreibung der Output-Struktur des get next job Statements

Jobserver Commands

multicommand

multicommand

Zweck

Zweck

Der Zweck des multicommands ist es mehrere SDMS-Kommandos als Einheit zuzuführen.

Syntax

Syntax

Die Syntax des multicommand Statements ist

begin multicommand commandlist end multicommand

begin multicommand commandlist end multicommand rollback

Beschreibung

Beschreibung

Mit den multicommands ist es möglich mehrere SDMS-Kommandos zusammen, d.h. in einer Transaktion aus zu führen. Damit wird gewährleistet, dass entweder alle Statements fehlerfrei ausgeführt werden, oder nichts passiert. Desweiteren wird die Transaktion nicht von anderen schreibenden Transaktionen unterbrochen. Wird das Keyword rollback spezifiziert, wird die Transaktion am Ende der Verarbeitung rückgängig gemacht. Auf dieser Weise kann getestet werden ob die Statements (technisch) korrekt verarbeitet werden können.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

reassure

Jobserver Commands

reassure

Zweck

Das reassure job Statement wird eingesetzt um eine Bestätigung über die Notwendig kwirck einen Job zu starten, nachdem ein Jobserver gestaret wurde, von dem Server zu bekommen.

Syntax

Die Syntax des reassure Statements ist

Syntax

reassure jobid [with run = integer]

Beschreibung

Mit dem reassure Statement bekommt man vom Server eine Bestätigung ob ein Job gestartet werden soll. Dieses Statement wird eingesetzt in dem Moment, dass ein Jobserver beim Hochfahren einen Job im Status **starting** vorfindet.

Beschreibung

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

reassure | 391

Jobserver Commands

register

register

Zweck

Zweck

Das register Statement wird eingesetzt um den Server zu benachrichtigen, dass der Jobserver bereit ist Jobs auszuführen.

Syntax

Syntax

Die Syntax des register Statements ist

```
register serverpath . servername
with pid = pid [ suspend ]
```

register with pid = pid

Beschreibung

Beschreibung

Die erste Form wird vom Operator benutzt um das Aktivieren von Jobs auf dem spezifizierten Jobserver zu ermöglichen.

Die zweite Form wird vom Jobserver selbst benutzt um den Server über seine Bereitschaft Jobs auszuführen zu informieren.

Unabhängig davon ob der Jobserver connected ist oder nicht, werden Jobs für diesen Server eingeplant, es sei den der Jobserver ist suspended.

Siehe das Statement 'deregister' auf Seite 150.

pid Die pid Option liefert dem Server Informationen über die Prozess ID des Jobservers auf Betriebsebene.

suspend Die suspend Option bewirkt, dass der Jobserver in den Suspended Zustand überführt wird.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Teil IV. Job Commands

alter job

alter job

Zweck

Zweck

Das alter job Statement wird benutzt um Eigenschaften des spezifizierten Jobs zu ändern. Es wird von den Job Administratoren, Jobservern und von dem Job selbst benutzt.

Syntax

Syntax Die Syntax des alter job Statements ist

```
alter job jobid
with WITHITEM {, WITHITEM}
alter job
with WITHITEM {, WITHITEM}
WITHITEM:
     < suspend | suspend restrict | suspend local | suspend local restrict >
     cancel
     clear warning
     comment = string
     error text = string
     exec pid = pid
     exit code = signed_integer
     exit state = statename [ force ]
     ext pid = pid
     ignore resource = ( id {, id} )
     ignore dependency = ( jobid [ recursive ] {, jobid [ recursive ]} )
     kill
     nicevalue = signed_integer
     priority = integer
     renice = signed_integer
     rerun [recursive]
     resume
     < noresume | resume in period | resume at datetime >
     run = integer
     state = JOBSTATE
     timestamp = string
     warning = string
```

JOBSTATE:

alter job

Job Commands

broken active
broken finished
dependency wait
error
finished
resource wait
running
started
starting
synchronize wait

Beschreibung

Das alter job Kommando wird für mehrere Zwecke genutzt. Als erstes verwenden Jobserver dieses Kommando um den Ablauf eines Jobs zu dokumentieren. Alle Statuswechsel eines Jobs während der Zeit in der der Job innerhalb der Zuständigkeit eines Jobservers fällt, werden mittels des alter job Kommandos ausgeführt. Zweitens werden einige Änderungen wie z. B. das Ignorieren von Abhängigkeiten oder Resourcen sowie das Ändern der Priorität eines Jobs manuell von einem Administrator ausgeführt.

Der Exit State eines Jobs in einem Pending State kann von dem Job selbst gesetzt werden, bzw. von einem Prozess welcher die Job ID und den Key des zu ändernden Jobs kennt.

cancel Die cancel Option wird benutzt um den adressierten Job und alle nicht Final Children zu canceln. Ein Job kann nur gecancelled werden wenn weder der Job selbst noch einer seiner Kinder aktiv ist.

Wenn ein Scheduling Entity von dem gecancelten Job abhängig ist, kann er unreachable werden. In diesem Fall erhält der abhängige Job nicht den im Exit State Profile definierten Unreachable Exit State, sondern wird in den Job Status "Unreachable" versetzt. Es ist Aufgabe des Operators diese Jobs nun mittels des Ignorierens von Abhängigkeiten wieder in den Job Status "Dependency Wait" zu versetzen, oder aber diese Jobs auch zu canceln.

Gecancelte Jobs werden wie Final Jobs ohne Exit State betrachtet. Das bedeutet, die Parents eines gecancelten Jobs werden final, ohne den Exit State des gecancelten Jobs zu berücksichtigen. Die abhängigen Jobs der Parents laufen in diesem Fall normal weiter.

Die cancel Option kann nur von Benutzern genutzt werden.

comment Die comment Option wird benutzt um eine Aktion zu dokumentieren oder um dem Job einen Comment zuzufügen. Comments können maximal 1024 Zeichen lang sein. Es kann eine beliebige Anzahl Comments für einen Job gespeichert werden.

Beschreibung

alter job

Einige Comments werden automatisch gespeichert. Wenn z. B. ein Job einen Restartable State erreicht, wird ein Protokoll geschrieben, um diesen Fakt zu dokumentieren.

error text Die error text Option wird benutzt um Fehlerinformation zu einem Job zu schreiben. Dieses kann von dem verantwortlichen Jobserver oder einem Benutzer gemacht werden. Der Server kann diesen Text auch selbst schreiben. Diese Option wird normalerweise benutzt, wenn der Jobserver den entsprechenden Prozess nicht Starten kann. Mögliche Fälle sind die Unmöglichkeit zum definierten Working Directory zu wechseln, die Unauffindbarkeit des ausführbaren Programmes oder Fehler beim Öffnen des Error Logfiles.

exec pid Die exec pid Option wird ausschließlich von dem Jobserver benutzt um die Prozess ID des Kontroll Prozesses innerhalb des Servers zu setzen.

exit code Die exit code Option wird von dem Jobserver benutzt um dem Repository Server mitzuteilen mit welchem Exit Code sich ein Prozess beendet hat. Der Repository Server berechnet jetzt den zugehörigen Exit State aus dem verwendeten Exit State Mapping.

exit state Die Exit State Option wird von Jobs in einem Pending State benutzt, um ihren State auf einen anderen Wert zu setzen. Dies wird normalerweise ein Restartable oder Final State sein.

Alternativ dazu kann diese Option von Administratoren benutzt werden, um den State von einem nonfinal Job zu setzen.

Sofern das Force Flag nicht benutzt wird, sind die einzigen States die gesetzt werden können, die States welche, durch die Anwendung des Exit State Mappings auf irgendeinem Exit Code, theoretisch erreichbar sind. Der gesetzte State muss im Exit State Profile vorhanden sein.

ext pid Die ext pid Option wird ausschließlich von dem Jobserver genutzt, um die Prozess ID des gestarteten Benutzerprozesses zu setzen.

ignore resource Die ignore resource Option wird benutzt um einzelne Resource Requests aufzuheben. Die ignorierte Resource wird nicht mehr beantragt.

Wenn Parameter einer Resource referenziert werden, kann diese Resource nicht ignoriert werden.

Wenn ungültige ID's spezifiziert wurden, wird dies übergangen. Alle anderen spezifizierten Resources werden ignoriert. Ungültige ID's in diesem Kontext sind ID's von Resources die von dem Job nicht beantragt werden.

Das Ignorieren von Resources wird protokolliert.

ignore dependency Die ignore dependency Option wird benutzt um definierte Dependencies zu ignorieren. Wenn das **recursive** flag benutzt wird, ignorieren nicht nur der Job oder Batch selbst, sondern auch seine Kinder die Dependencies.

kill Die kill Option wird benutzt um den definierten Kill Job zu submitten. Wenn kein Kill Job definiert ist, ist es nicht möglich den Job vom BICsuite aus erzwungenermaßen zu terminieren. Natürlich muss der Job aktiv sein, das bedeutet, der Jobstate muss **running**, **killed** oder **broken_active** sein. Die letzten beiden States sind keine regulären Fälle.

Wenn ein Kill Job submitted wurde, ist der Jobstate **to_kill**. Nachdem der Kill Job beendet wurde, wird der Jobstate des killed Jobs in den State **killed** gesetzt, es sei denn er ist beendet, dann wird der Jobstate **finished** oder **final** sein. Das bedeutet, der Job mit dem Jobstate **killed** bedeutet, dass er immer noch running ist und dass mindestens ein Versuch gemacht wurde, den Job zu terminieren.

nicevalue Die nicevalue Option wird benutzt um die Priorität oder den Nicevalue eines Jobs oder Batches und allen ihren Children zu ändern. Hat ein Child mehrere Parents, kann eine Änderung, muss aber nicht, in dem Nicevalue von einem der Parents Auswirkungen auf die Priorität des Childs haben. In dem Fall das es mehrere Parents gibt wird das maximale Nicevalue gesucht.

Also, wenn Job C drei Parents P1, P2 und P3 hat und P1 setzt einen Nicevalue von 0, P2 einen von 10 und P3 einen von -10, ist der effektive Nicevalue -10. (Umso niederiger der Nicevalue, umso besser). Wenn der Nicevalue von P2 auf -5 geändert wird, passiert nichts, weil die -10 von P3 besser als -5 ist. Wenn jetzt der Nicevalue von P3 auf 0 sinkt, wird die neue effektive Nicevalue für Job C -5.

Die Nicevalues können Werte zwischen -100 und 100 haben. Werte die diese Spanne übersteigen, werden stillschweigend angepasst.

priority Die priority Option wird benutzt, um die (statische) Priorität eines Jobs zu ändern. Weil Batches und Milestones nicht ausgeführt werden, haben Prioritäten keine Bedeutung für sie.

Ein Wechsel der Priorität betrifft nur den geänderten Job. Gültige Werte liegen zwischen 0 und 100. Dabei korrespondiert 100 mit der niedrigsten Priorität und 0 mit der höchsten Priorität.

Bei der Berechnung der dynamischen Priorität eines Jobs, startet der Scheduler mit der statischen Priorität und passt dies, entsprechend der Zeit in der der Job schon wartet, an. Wenn mehr als ein Job die gleiche dynamische Priorität hat, wird der Job mit der niedrigsten Job ID als erster gescheduled.

renice Die renice Option gleicht der nicevalue Option mit dem Unterschied, das die renice Option relativ arbeitet, während die nicevalue Option absolut arbeitet. Wenn einige Batches einen Nicevalue von 10 haben bewirkt eine renice von -5,

alter job

dass die Nicevalue auf 5 zunimmt. (Zunahme, weil je niedriger die Nummer, desto höher die Piorität)

rerun Die rerun Option wird benutzt um einen Job in einem Restartable State neu zu starten. Der Versuch einen Job, der nicht restartable ist, neu zu starten, führt zu einer Fehlermeldung. Ein Job ist restartable, wenn er in einem Restartable State oder in einem **error** oder **broken_finished** Jobstate ist.

Wenn das **recursive** flag spezifiziert ist, wird der Job selbst und alle direkten und indirekten Children, die in einem Restartable State sind, neu gestartet. Wenn der Job selbst Final ist, wird das in dem Fall nicht als Fehler betrachtet. Es ist also möglich Batches rekursiv neu zu starten.

resume Die resume Option wird benutzt um einen Suspended Job oder Batch zu reaktivieren. Es gibt dabei zwei Möglichkeiten. Erstens kann der suspended Job oder Batch sofort reaktiviert werden, und zweitens kann eine Verzögerung eingestellt werden.

Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll.

Für die Spezifikation einer Zeit siehe auch die Übersicht auf Seite 6.

Die resume Option kann zusammen mit der suspend Option verwendet werden. Dabei wird der Job suspended und nach der (bzw. zur) spezifizierten Zeit wieder resumed.

run Die run Option wird von dem Jobserver benutzt zwecks der Sicherstellung, dass der geänderte Job mit der aktuellen Version übereinstimmt.

Theoretisch ist es möglich, dass nachdem ein Job von einem Jobserver gestartet wurde, der Computer abstürzt. Um die Arbeit zu erledigen wird der Job mittels eines manuellen Eingriffs, von einem anderen Jobserver, neu gestartet. Nach dem hochfahren des ersten Systems, kann der Jobserver versuchen den Jobstate nach broken_finished zu ändern, ohne über das Geschehen nach dem Absturz bescheid zu wissen. Das Benutzen der run Option, verhindert nun das fälschlich Setzen des Status.

state Die state Option wird hauptsächlich von Jobservern benutzt, sie kann aber auch von Administratoren benutzt werden. Es wird nicht empfohlen das so zu machen, es sei denn Sie wissen genau was Sie tun.

Die übliche Prozedur ist, das der Jobserver den State eines Jobs von **starting** nach **started**, von **started** nach **running** und von **running** nach **finished** setzt. Im Falle eines Absturzes oder anderen Problemen ist es möglich dass der Jobserver einen Job in einen **broken_active** oder **broken_finished** State setzt. Das bedeutet, der Exit Code von dem Prozess steht nicht zur Verfügung und der Exit State muss manuell gesetzt werden.

alter job

Job Commands

suspend Die suspend Option wird benutzt um einen Batch oder Job zu suspendieren. Sie arbeitet immer rekursiv. Wenn ein Parent Suspended ist, sind auch alle Children suspended. Die resume Option wird benutzt um die Situation zu umzukehren.

Die **restrict** Option bewirkt, dass nur Benutzer der ADMIN Gruppe die Suspendierung wieder aufheben können.

timestamp Die timestamp Option wird von dem Jobserver benutzt um die Timestamps der Statuswechsel zu setzen, gemäß der lokalen Zeit aus Sicht des Jobservers.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

connect

connect

Zweck

Zweck

Das connect Statement wird eingesetzt um einen Job bei dem Server zu authentifizieren.

Syntax

Syntax

Die Syntax des connect Statements ist

connect job jobid identified by string [with WITHITEM {, WITHITEM}]

```
WITHITEM:
```

```
command = ( sdms-command )
| protocol = PROTOCOL
| session = string
| timeout = integer
| < trace | notrace >
| trace level = integer
```

PROTOCOL:

json | line | perl | python

serial xml

Beschreibung

Beschreibung

Das connect Kommando wird benutzt um den verbundenen Prozess beim Server zu authentifizieren. Es kann wahlweise ein Kommunikationsprotokoll spezifiziert werden. Das default protocol ist **line**.

Das ausgewählte Protokoll definiert die Form des Outputs. Alle Protokolle, außer serial, liefern ASCII Output. Das Protokoll serial liefert einen serialized Java Objekt zurück.

Beim Connect kann gleich ein auszuführendes Kommando mitgegeben werden. Als Output des Connect Kommandos wird in dem Fall der Output des mitgegebenen Kommandos genutzt. Falls das Kommando fehl schlägt, der Connect aber gültig war, bleibt die Connection bestehen.

Im folgenden ist für alle Protokolle, außer für das **serial** Protokoll, ein Beispiel gegeben.

402 | connect

connect Job Commands

Das line protocol Das line protocol liefert nur einen ASCII Text als Ergebnis von einem Kommando zurück.

```
connect donald identified by 'duck' with protocol = line;
Connect
CONNECT_TIME : 19 Jan 2005 11:12:43 GMT
Connected
SDMS>
```

Das XML protocol Das XML protocol liefert eine XML-Struktur als Ergebnis eines Kommandos zurück.

```
connect donald identified by 'duck' with protocol = xml;
<OUTPUT>
<DATA>
<TITLE>Connect</TITLE>
<RECORD>
<CONNECT_TIME>19 Jan 2005 11:15:16 GMT</CONNECT_TIME></RECORD>
</DATA>
<FEEDBACK>Connected</FEEDBACK>
</OUTPUT>
```

Das python protocol Das python protocol liefert eine python-Struktur, welche durch die python eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = python;
{
'DATA':
{
'TITLE': 'Connect',
'DESC': [
'CONNECT_TIME'
],
'RECORD': {
'CONNECT_TIME': '19 Jan 2005 11:16:08 GMT'}
}
,'FEEDBACK': 'Connected'
}
```

connect | 403

connect

Das perl protocol Das perl protocol liefert eine perl Struktur, welche durch die perl eval Funktion ausgewertet werden kann, zurück.

```
connect donald identified by 'duck' with protocol = perl;
{
'DATA' => {
   'TITLE' => 'Connect',
   'DESC' => [
   'CONNECT_TIME'
],
   'RECORD' => {
   'CONNECT_TIME' => '19 Jan 2005 11:19:19 GMT'}
}, 'FEEDBACK' => 'Connected'
}
```

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

404 l connect

disconnect Job Commands

disconnect

Zweck

Das disconnect Statement wird eingesetzt um die Serververbindung zu beenden. Zweck

Syntax

Die Syntax des disconnect Statements ist

Syntax

disconnect

Beschreibung

Mit dem disconnect Statement kann die Verbindung zum Server beendet wer- Beschreibung den.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

get parameter

get parameter

Zweck

Zweck

Das get parameter Statement wird eingesetzt um den Wert des spezifizierten Parameters innerhalb des Kontext des anfordernden Jobs, entsprechend dem spezifizierten Jobs, zu bekommen.

Syntax

Syntax

Die Syntax des get parameter Statements ist

```
get parameter parametername [ < strict | warn | liberal > ]
get parameter of jobid parametername [ < strict | warn | liberal > ]
```

Beschreibung

Beschreibung

Das get parameter-Statement wird eingesetzt um den Wert des spezifierten Parameters innerhalb des Kontextes eines Jobs zu bekommen.

Die Zusatzoption hat dabei folgende Bedeutung:

	-	
Option	Bedeutung	
strict	Der Server liefert einen Fehler, wenn der gefragte Parameter nicht ex-	
	plizit in der Job Definition deklariert ist	
warn	Es wird eine Meldung ins Logfile des Server geschrieben, wenn ver-	
	sucht wird den Wert eines nicht deklarierten Parameters zu ermitteln.	
liberal	Der Versuch nicht deklarierte Parameter ab zu fragen wird	
	stillschweigend erlaubt.	
Dan Dafas	alternation alternation of	

Das Defaultverhalten hängt von der Serverkonfiguration ab.

Ausgabe

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
VALUE	Wert des angeforderten Parameters

Tabelle 28.1.: Beschreibung der Output-Struktur des get parameter Statements

get submittag

Job Commands

get submittag

Zweck

Das get submittag Statement wird eingesetzt um eine eindeutige Identifikation Z von dem Server zu bekommen. Diese Identifikation kann benutzt werden, um race conditions zwischen frontend und backend während des Submits zu verhindern.

Zweck

Syntax

Die Syntax des get submittag Statements ist

Syntax

get submittag

Beschreibung

Mit dem get submittag Statement bekommt man eine Identifikation von dem Server. Damit verhindert man race conditions zwischen frontend und backend wenn Jobs submitted werden.

Beschreibung

Eine solche Situation entsteht, wenn aufgrund eines Fehlers die Rückmeldung des Submits nicht ins Frontend eintrifft. Durch Benutzung eines Submittags kann das Frontend gefahrlos einen zweiten Versuch starten. Der Server erkennt ob der betreffende Job bereits submitted wurde und antwortet dementsprechend. Ein doppeltes Submitten des Jobs wird damit zuverlässig verhindert.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
VALUE	Das angeforderte Submit Tag

Tabelle 28.2.: Beschreibung der Output-Struktur des get submittag Statements

multicommand

multicommand

Zweck

Zweck

Der Zweck des multicommands ist es mehrere SDMS-Kommandos als Einheit zuzuführen.

Syntax

Syntax

Die Syntax des multicommand Statements ist

begin multicommand commandlist end multicommand

begin multicommand commandlist end multicommand rollback

Beschreibung

Beschreibung

Mit den multicommands ist es möglich mehrere SDMS-Kommandos zusammen, d.h. in einer Transaktion aus zu führen. Damit wird gewährleistet, dass entweder alle Statements fehlerfrei ausgeführt werden, oder nichts passiert. Desweiteren wird die Transaktion nicht von anderen schreibenden Transaktionen unterbrochen. Wird das Keyword rollback spezifiziert, wird die Transaktion am Ende der Verarbeitung rückgängig gemacht. Auf dieser Weise kann getestet werden ob die Statements (technisch) korrekt verarbeitet werden können.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

set parameter

Job Commands

set parameter

Zweck

Das set parameter Statement wird eingesetzt um den Wert des spezifizierten Zweck Parameters innerhalb des Kontext eines Jobs zu setzen.

Syntax

Die Syntax des set parameter Statements ist

Syntax

```
set parameter parametername = string {, parametername = string}
set parameter < on | of > jobid parametername = string {,
parametername = string}
set parameter < on | of > jobid parametername = string {,
parametername = string} identified by string
```

Beschreibung

Mittels des set parameter Statements können Jobs oder Benutzer Parameterwerte Beschreibung in dem Kontext des Jobs setzen.

Falls die **identified by** Option spezifiziert ist, wird der Parameter nur dann gesetzt, wenn das Paar *jobid* und *string* eine Anmeldung ermöglichen wurden.

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

Ausgabe

set state

set state

Zweck

Zweck

Das set state Statement wird eingesetzt um den Exit State eines Jobs in einem pending Exit State zu setzen.

Syntax

Syntax

Die Syntax des set state Statements ist

set state = *statename*

Beschreibung

Beschreibung

Das set state Statement wird eingesetzt um den Exit State eines Jobs in einem pending Exit State zu setzen.

Ausgabe

Ausgabe

Es wird die Bestätigung einer erfolgreichen Durchführung zurückgeliefert.

410 | set state

submit Job Commands

submit

Zweck

Das submit Statement wird eingesetzt um einen Master Batch oder Job, sowie Zweck alle definierten Children, auszuführen.

Syntax

```
Die Syntax des submit Statements ist

Syntax

submit folderpath [ with WITHITEM {, WITHITEM} ]
```

Beschreibung

Das submit Statement wird benutzt um einen Job oder Batch zu submitten. Es Beschreibung existieren zwei Formen des Submit-Kommandos.

- Die erste Form wird von Benutzern, welche auch Programme sein können und dem Time Scheduling Module genutzt. Diese Form submitted Master Jobs und Batches.
- Die zweite Form des Statements wird von Jobs genutzt, um dynamische Children zu submitten.

submit | 411

submit

check only Die check only Option wird benutzt um zu überprüfen ob ein Master Submittable Batch oder Job submitted werden kann. Das bedeutet, es wird geprüft ob alle Abhängigkeiten erfüllt werden können und alle referenzierten Parameter definiert sind.

Es wird nicht überprüft ob die Jobs in irgendeinem Scope ausgeführt werden können oder nicht. Das ist eine Situation die jederzeit zur Laufzeit auftreten kann.

Eine positive Rückmeldung bedeutet, dass der Job oder Batch aus Sicht des Systems submitted werden kann.

Die check only Option kann nicht in einem Job Kontext benutzt werden.

childtag Die childtag Option wird von Jobs benutzt, um verschiedene Instanzen von dem selben Scheduling Entity zu submitten und um zwischen ihnen unterscheiden zu können.

Es führt zu einem Fehler, wenn der gleiche Scheduling Entity doppelt submitted wird, wenn sich der Childtag nicht unterscheidet. Der Inhalt des Childtags hat keine weitere Bedeutung für den Scheduling System.

Die maximale länge eines Childtags beträgt 70 Zeichen. Die Childtag Option wird im Falle eines Master Submits ignoriert.

group Die group Option wird benutzt um die Owner-Gruppe auf den spezifizierten Wert zu setzen. Der Benutzer muss zu dieser Gruppe gehören, es sei denn er gehört zu der priviligierten Gruppe ADMIN, in diesem Fall kann jede beliebige Gruppe spezifiziert werden.

nicevalue Die nicevalue Option definiert eine Korrektur die für die Berechnung der Prioritäten des Jobs und seinen Children benutzt wird. Es sind Werte von -100 bis 100 erlaubt.

parameter Die parameter Option wird benutzt um den Wert von Job Parameter beim Submit zu spezifizieren. Die Parameter werden in dem Scope des Master Batches oder Jobs gesetzt. Das Bedeutet, wenn Parameter, die nicht in dem Master Batch oder Job definiert sind, spezifiziert werden, sind diese Parameter unsichtbar für Children.

submittag Wenn der submittag spezifiziert ist, muss er eine eindeutige Bezeichnung für den Submitted Entity haben. Dieser Tag wurde, um imstande zu sein Jobs und Batches programmatisch zu submitten und um den Job oder Batch, mit demselben Tag, nach einem Absturz von einem der Komponenten neu zu submitten. Wenn die Submission des Jobs das erste mal erfolgreich war, wird der zweite Submit einen Fehler melden. Wenn nicht, wird der zweite Submit erfolgreich sein.

412 I submit

submit Job Commands

unresolved Die unresolved Option spezifiziert wie der Server bei nicht auflösbaren Abhängigkeiten reagieren sollte. Diese Option wird hauptsächlich benutzt, wenn Teile eines Batches nach Reparaturarbeiten submitted werden. Der Fehlerhafte Teil wird normal gecancelt und dann als Master Run neu submittet. Die vorherigen Abhängigkeiten müssen in diesem Fall ignoriert werden, andernfalls wird der Submit scheitern.

suspend Die suspend Option wird benutzt um Jobs oder Batches zu submitten und sie zur selben Zeit zu suspenden. Wenn nichts festgelegt wurde, wird nicht suspended. Dies kann explizit zur Submit-Zeit spezifiziert werden. Wenn ein Job oder Batch suspended wurde wird er, sowie auch seine Children, nicht gestartet. Wenn ein Job bereits läuft, wird er keinen Final State erreichen, wenn er suspended ist.

resume Die resume Option kann zusammen mit der suspend Option verwendet werden um eine verzögerte Ausführung zu bewirken. Es gibt dabei zwei Möglichkeiten. Entweder erreicht man eine Verzögerung dadurch, dass die Anzahl von Zeiteinheiten die gewartetet werden sollen, spezifiziert werden, oder aber man spezifiziert den Zeitpunkt zu dem der Job oder Batch aktiviert werden soll. Mit dieser Option kann die at-Funktionalität ohne das Anlegen eines Schedules nachgebildet werden.

Ausgabe

Dieses Statement liefert eine Output-Struktur vom Typ Record.

Ausgabe

Output Beschreibung Die Datenelemente des Outputs werden in der untenstehenden Tabelle beschrieben.

Feld	Beschreibung
ID	Id des Submitted Entities

Tabelle 28.3.: Beschreibung der Output-Struktur des submit Statements

submit | 413