

CA Workload Automation AE

Reference Guide

r11.3



This documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2010 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Access Control
- CA AutoSys Workload Automation Connect Option (CA AutoSys WA Connect Option)
- CA Embedded Entitlements Manager (CA EEM)
- CA Job Management Option
- CA Jobtrac™ Job Management (CA Jobtrac JM)
- CA Network and Systems Management (CA NSM)
- CA NSM Event Management
- CA NSM Management Command Center (CA NSM MCC)
- CA Scheduler® Job Management (CA Scheduler JM)
- CA Service Desk
- CA Spectrum Automation Manager (formerly named CA DCA Manager)
- CA Universal Job Management Agent (CA UJMA)
- CA Workload Automation AE (formerly named CA AutoSys Workload Automation)
- CA Workload Automation Agent for UNIX (CA WA Agent for UNIX)
- CA Workload Automation Agent for Linux (CA WA Agent for Linux)
- CA Workload Automation Agent for Windows (CA WA Agent for Windows)
- CA Workload Automation Agent for i5/OS (CA WA Agent for i5/OS)
- CA Workload Automation Agent for Application Services (CA WA Agent for Application Services)
- CA Workload Automation Agent for Web Services (CA WA Agent for Web Services)
- CA Workload Automation Agent for Databases (CA WA Agent for Databases)
- CA Workload Automation Agent for SAP (CA WA Agent for SAP)
- CA Workload Automation Agent for PeopleSoft (CA WA Agent for PeopleSoft)
- CA Workload Automation Agent for Oracle E-Business Suite (CA WA Agent for Oracle E-Business Suite)
- CA Workload Automation Agent for z/OS (CA WA Agent for z/OS)
- CA Workload Automation EE (formerly named CA ESP Workload Automation)
- CA Workload Automation SE (formerly named CA 7 Workload Automation)

- CA Workload Control Center (CA WCC)
- CA Desktop and Server Management (CA DSM)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA Technologies products. At <http://ca.com/support>, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA Technologies product documentation, complete our short customer survey, which is available on the CA Support website at <http://ca.com/docs>.

Contents

Chapter 1: Introduction	23
Intended Audience	23
Legacy Agent Replaced by CA Workload Automation Agent	23
Chapter 2: Commands	25
JIL Syntax Rules	25
Issuing Commands on UNIX	27
Issuing Commands on Windows	28
Wildcard Characters in Commands	28
Commands by Task	29
archive_events Command—Archive Data	30
Archive Files	33
archive_jobs Command—Delete Obsolete Job Versions	36
as_config Command—Manage Encryption Keys and Passwords	37
as_info Command—Return Installation Information	38
as_safetool Command—Maintain Authentication Certificates	40
Examples: Running as_safetool in Interactive Mode	42
Exit Codes for the as_safetool Command	46
as_server Command—Run the Application Server	47
as_test Command—Test Utility	48
astail Command—Display End of File	49
auto_svcdesk Command—Opens CA Service Desk Tickets	50
autoaggr Command—Aggregate Data	54
autocal_asc Command—Manage Calendars	55
Standard Calendars	61
Extended Calendars	61
Cycles	69
autoflags Command—Return System Information	69
autoping Command—Verify Server, Agent, and Client Communication	71
autoprofm Command—Convert Job Profiles to the New Format (Windows Only)	73
autorep Command—Report Job, Machine, and Variable Information	75
autorep Command Usage Notes	81
Types of autorep Reports	82
Columns in an autorep Report	82
Examples: Generate Reports Using autorep	85
autosec_test Command—Simulate a Call to the Security Subsystem	96

autostatus Command—Report Job Status or Global Variable Value	100
autostatad Command—Report the Status of a CA AutoSys Workload Automation Adapter Job	105
autosys_secure Command—Define Security Settings	107
Native and Embedded Security	110
EDIT Superuser and EXEC Superuser	110
Database Password	111
Remote Authentication Method	111
User and Host Management	111
Encrypted Passwords	112
Running autosys_secure	112
autosyslog Command—Display the Scheduler, Application Server, and Log File for a Job	114
autotimezone Command—Manage the ujo_timezones Table	118
TZ Variable Syntax	121
autotrack Command—Tracks Changes to the Database	126
chase Command—Verify Job STARTING or RUNNING Status	135
Running chase Automatically	137
chk_auto_up Command—Confirm Application Server, Scheduler, and Event Server Status	138
Exit Codes	140
clean_files Command—Delete Old Agent Log Files from Legacy Agents	155
cron2jil Command—Convert UNIX crontab Data to a JIL Script or Calendar File	157
dbspace Command—Calculate and Report Database Table Space	159
DBMaint Command—Maintain the CA Workload Automation AE Database	160
dbstatistics Command—Generate Data Server Statistics	162
event_demon Command—Run CA Workload Automation AE	163
eventor Command—Start the Scheduler	164
Log Files	165
forecast Command—Report Job Flow	166
initautosys Command—Open the Instance Command Prompt Window	169
jil Command—Run the Job Information Language Processor	170
JIL Syntax Rules	173
job_depends Command—Generate Detailed Reports of Job Dependencies and Conditions	176
monbro Command—Run a Monitor or Report	180
sendevent Command—Send Events	183
time0 Command—Calculate Internal CA Workload Automation AE Time	198

Chapter 3: Job Types 201

Job Types	201
Box Jobs	201
Command Jobs	204
CPU Monitoring Jobs	206
Database Monitor Jobs	207

Database Stored Procedure Jobs	209
Database Trigger Jobs	211
Disk Monitoring Jobs.....	213
Entity Bean Jobs	214
File Trigger Jobs	217
File Watcher Jobs	219
FTP Jobs	220
HTTP Jobs	222
i5/OS Jobs	224
IP Monitoring Jobs.....	226
JMS Publish Jobs	228
JMS Subscribe Jobs	230
JMX-MBean Attribute Get Jobs	232
JMX-MBean Attribute Set Jobs.....	233
JMX-MBean Create Instance Jobs	235
JMX-MBean Operation Jobs	237
JMX-MBean Remove Instance Jobs	239
JMX-MBean Subscribe Jobs	240
Oracle E-Business Suite Copy Single Request Jobs	242
Oracle E-Business Suite Request Set Jobs	244
Oracle E-Business Suite Single Request Jobs.....	246
PeopleSoft Jobs	249
POJO Jobs	252
Process Monitoring Jobs	253
RMI Jobs	254
SAP Batch Input Session Jobs	255
SAP BW InfoPackage Jobs	256
SAP BW Process Chain Jobs	258
SAP Data Archiving Jobs	259
SAP Event Monitor Jobs	261
SAP Job Copy Jobs	262
SAP Process Monitor Jobs	264
SAP R/3 Jobs	266
Secure Copy Jobs	268
Session Bean Jobs	270
SQL Jobs	272
Text File Reading and Monitoring Jobs	274
Web Service Jobs	276
Windows Event Log Monitoring Jobs.....	279
Windows Service Monitoring Jobs.....	281
z/OS Data Set Trigger Jobs	282
z/OS Manual Jobs	284

z/OS Regular Jobs	285
-------------------------	-----

Chapter 4: JIL Job Definitions 287

delete_box Subcommand—Delete a Box Job from the Database	287
delete_job Subcommand—Delete a Job from the Database	288
insert_job Subcommand—Add a Job to the Database	289
override_job Subcommand—Define a Job Override	290
update_job Subcommand—Update an Existing Box or Job Definition in the Database	292
Common Job Attributes	293
alarm_if_fail Attribute—Specify Whether to Post an Alarm for FAILURE or TERMINATED Status	295
application Attribute—Associate a Job with an Application	296
arc_obj_name Attribute—Identify Name of SAP Archiving Object	297
arc_obj_variant Attribute—Identify Name of SAP Archiving Object Variant	298
arc_parms Attribute—Specify Archive Parameters	299
auth_string Attribute—Define an Authorization String	303
auto_delete Attribute—Automatically Delete a Job on Completion	304
auto_hold Attribute—Put a Job on Hold when Its Container’s Status Changes to RUNNING	305
avg_runtime Attribute—Define Average Run Time for a New Job	306
bdc_err_rate Attribute—Specify the Maximum Acceptable Error Rate	307
bdc_ext_log Attribute—Generate Advanced Logging of the Batch Input Session	308
bdc_proc_rate Attribute—Specify Minimum Acceptable Process Rate	309
bdc_system Attribute—Specify the Name of a Background Server	310
bean_name Attribute—Specify the JNDI Name of the Bean	311
blob_file Attribute—Specify File for the Blob	312
blob_input Attribute—Specify Input for the Blob	313
box_failure Attribute—Define Criteria for Box Job Failure	314
box_name Attribute—Identify a Box as Container for a Job	316
box_success Attribute—Define Criteria for Box Job Success	317
box_terminator Attribute—Terminate Box on Job Failure	319
chk_files Attribute—Verify File Space Required to Start a Job	320
class_name Attribute—Specify a Class Name	322
command Attribute—Specify a Command or Script to Run	323
UNIX Considerations for the command Attribute	325
Examples: Specifying the command Attribute on UNIX	328
Windows Considerations for the command Attribute	330
Examples: Specifying the command Attribute on Windows	333
condition Attribute—Define Starting Conditions for a Job	335
Job Status Dependencies	337
Cross-Instance Job Dependencies	340
Exit Code Dependencies	342
Global Variable Dependencies	343

Look-Back Dependencies	344
condition_code Attribute—Specify a Code to Indicate Success or Failure	346
connect_string Attribute—Specify Database Resource Location	349
connection_factory Attribute—Specify the JNDI Name of the Connection Factory	351
continuous Attribute—Specify Whether the Job Monitors Continuously	353
copy_jcl Attribute—Specify a Data Set Name that Receives the Copy JCL	356
create_method Attribute—Specify a Create Method	357
create_name Attribute—Specify a Create Method	358
create_parameter Attribute—Specify Create Parameters	359
cpu_usage Attribute—Specify Whether to Monitor Available or Used CPU	360
date_conditions Attribute—Base Job Start on Date and Time Attribute Values	362
days_of_week Attribute—Specify which Days of the Week to Run a Job	363
dbtype Attribute—Specify the Type of Database	365
description Attribute—Define a Text Description for a Job	366
destination_file Attribute—Specify an Output Destination File	367
destination_name Attribute—Specify the JNDI Name of the Topic or Queue	369
disk_drive Attribute—Specify the Disk Space to Monitor	371
disk_format Attribute—Specify the Unit of Measurement Used to Monitor Disk Space	372
disk_space Attribute—Specify Whether to Monitor Available or Used Disk Space	374
encoding Attribute—Specify the Character Encoding of a File	375
endpoint_URL Attribute—Specify a Target Endpoint URL	376
envvars Attribute—Specify a Job's Environment Variables	378
exclude_calendar Attribute—Define Days to Exclude a Job from Running	380
fail_codes Attribute—Define Exit Codes to Indicate Job Failure	381
filter Attribute—Specify a Filter Using Regular Expression Logic	383
filter_type Attribute—Specify the Filter Type	385
finder_name Attribute—Specify a Finder Method	386
finder_parameter Attribute—Specify Finder Parameters	387
ftp_command—Specify Site-specific FTP Commands	389
ftp_compression Attribute—Specify the Data Compression Level	390
ftp_local_name Attribute—Specify Local Filenames within an FTP Job	392
ftp_local_user Attribute—Specify the Local User ID on the Agent	395
ftp_remote_name Attribute—Specify Remote Filename within an FTP Job	396
ftp_server_name Attribute—Specify FTP Server Name within an FTP Job	400
ftp_server_port Attribute—Specify the Server Port of a Remote FTP Server	402
ftp_transfer_direction Attribute—Specify Transfer Direction within an FTP Job	403
ftp_transfer_type Attribute—Specify Data Code Type within an FTP Job	404
ftp_use_SSL Attribute—Specify SSL FTP or Regular FTP	406
group Attribute—Associate a Job with a Group	409
heartbeat_interval Attribute—Set the Monitoring Frequency for a Job	410
i5_action Attribute—Specify Whether to Run a Program or Issue a Command	411
i5_cc_exit Attribute—Specify the Type of Exit Code to Return	413

i5_curr_lib Attribute—Specify the Name of the Current Library	415
i5_job_desc Attribute—Specify the Job Description	416
i5_job_name Attribute—Specify the Job Name	417
i5_job_queue Attribute—Specify the Job Queue for a Program	418
i5_lda Attribute—Specify the Data for the Local Data Area	419
i5_lib Attribute—Specify the Name of a Library	420
i5_library_list Attribute—Specify the Libraries for an i5/OS Job	421
i5_name Attribute—Specify a CL Program, Source File, or Command	423
i5_others Attribute—Pass Keyword Parameters to SBMJOB	426
i5_params Attribute—Pass Positional Parameters	427
i5_process_priority Attribute—Specify the Process Priority of an i5/OS Job	428
initial_context_factory Attribute—Specify an Initial Context Factory	429
inside_range Attribute—Specify Whether to Monitor for CPU Usage Within or Outside a Range	431
interactive Attribute—Specify Whether to Run a Command Job in Interactive Mode on Windows	433
invocation_type Attribute—Specify the HTTP Method Type	434
ip_host Attribute—Specify the IP Address to Monitor	435
ip_port Attribute—Specify the Port Number at the IP Address to Monitor	437
ip_status Attribute—Specify the Status of the IP Address to Monitor	438
j2ee_authentication_order Attribute—Specify Authentication Protocols	439
j2ee_conn_domain Attribute—Specify a Domain for NTLM Connection Authentication	440
j2ee_conn_origin Attribute—Specify an Origin Host Name for NTLM Connection Authentication	442
j2ee_no_global_proxy_defaults Attribute—Specify Whether to Use Global Proxy Defaults	443
j2ee_parameter Attribute—Specify Input Parameters	444
j2ee_proxy_domain Attribute—Specify a Domain for Proxy Authentication	446
j2ee_proxy_host Attribute—Specify a Proxy Host	447
j2ee_proxy_origin_host Attribute—Specify an Origin Host Name for Proxy Authentication	449
j2ee_proxy_port Attribute—Specify a Proxy Port	450
j2ee_proxy_user Attribute—Specify a User for Proxy Authentication	451
j2ee_user Attribute—Specify a JNDI User Name	452
jcl_library Attribute—Specify a JCL Library	454
jcl_member Attribute—Specify a JCL Member	455
jmx_parameter Attribute—Specify JMX Parameters	456
jmx_user Attribute—Specify a JMX User Name	457
job_class Attribute—Assign a Job Class to a Job	458
job_load Attribute—Define a Job’s Relative Processing Load	460
job_terminator Attribute—Kill a Job if Its Box Fails	461
job_type Attribute—Specify Job Type	464
lower_boundary Attribute—Define the Start of the Range to be Searched (OMTF Jobs)	467
lower_boundary Attribute—Define the Minimum CPU or Disk Usage to Monitor (OMCPU and OMD Jobs)	470
machine Attribute—Define the Client Where a Job Runs	473
max_exit_success Attribute—Specify Maximum Exit Code for Success	475
max_run_alarm Attribute—Define the Maximum Run Time for a Job	476

mbean_attr Attribute—Specify the MBean Attribute to Query or Set	477
mbean_name Attribute—Specify the Name of the MBean	478
mbean_operation Attribute—Specify the Operation to be Invoked	480
message_class Attribute—Specify the Java Class of the JMS Message	481
method_name Attribute—Specify the Method or Operation to be Invoked	482
min_run_alarm Attribute—Define the Minimum Run Time for a Job	484
modify_parameter Attribute—Specify Modify Parameters	485
monitor_cond Attribute—Specify a Condition to Monitor For	487
monitor_mode Attribute—Specify Whether to Monitor Conditions Immediately or Continuously	488
monitor_type Attribute—Specify the Type of Database Change to Monitor For	492
must_complete_times Attribute—Specify the Time a Job Must Complete By	493
must_start_times Attribute—Specify the Time a Job Must Start By	499
n_retrys Attribute—Define the Number of Times to Restart a Job After a Failure	505
no_change Attribute—Define the Minimum Change Required in CPU or Disk Usage	507
notification_id Attribute—Identify the Recipient of the Notification	509
notification_msg Attribute—Define the Message to Include in the Notification	510
one_way Attribute—Specify Whether to Invoke the Service One Way	510
operation_type Attribute—Specify the Operation Type	512
oracle_appl_name Attribute—Specify the Name of an Oracle Applications Application	513
oracle_appl_name_type Attribute—Specify the Type of Oracle Applications Name	515
oracle_args Attribute—Define Argument Values to Pass to a Single Request	516
oracle_desc Attribute—Describe a Single Request Job	518
oracle_mon_children Attribute—Specify Whether to Monitor Children Programs	519
oracle_mon_children_delay Attribute—Define the Time to Wait Before Monitoring Children Programs	520
oracle_print_copies Attribute—Specify the Number of Copies to Print	522
oracle_print_style Attribute—Specify a Print Style	523
oracle_printer Attribute—Specify a Printer	525
oracle_program Attribute—Specify the Name of a Single Request Program	527
oracle_programdata Attribute—Specify Data for a Program in a Request Set	528
oracle_req_set Attribute—Specify the Name of a Request Set	530
oracle_resp Attribute—Specify a Responsibility Name	531
oracle_save_output Attribute—Specify Whether to Save Output	533
oracle_use_arg_def Attribute—Specify Whether to Use Argument Defaults	534
oracle_user Attribute—Specify a User Name	536
owner Attribute—Define the Owner of the Job	538
Changing the Owner in Multiple Jobs	542
Job Owners and Security	542
permission Attribute—Specify the Users with Edit and Execute Permissions	543
User Types	544
Permission Types on UNIX	545
Permission Types on Windows	546
User and Permission Types on UNIX	546

Job Permissions and Windows	547
poll_interval Attribute—Define the Frequency to Poll CPU or Disk Usage	547
port_name Attribute—Specify the Port Name Within the Namespace	549
priority Attribute—Define the Queue Priority of the Job	551
process_name Attribute—Specify the Name of the Process to be Monitored	552
process_status Attribute—Specify the Status of the Process to be Monitored	555
profile Attribute—Specify a Job Profile	557
provider_url Attribute—Specify a Service Provider URL (EJB and JMS Jobs)	560
provider_url Attribute—Specify a Host URL (HTTP Jobs)	563
ps_args Attribute—Pass Additional Parameters for the PeopleSoft Report	565
ps_dest_format Attribute—Specify the Output Format for a PeopleSoft Report	566
ps_dest_type Attribute—Specify the Output Type for a PeopleSoft Report	568
ps_detail_folder Attribute—Specify the Name of a PeopleSoft Distribution Detail Folder	570
ps_dlist_roles Attribute—Specify a Distribution List of Roles	571
ps_dlist_users Attribute—Specify a Distribution List of Operator IDs	572
ps_email_address Attribute—Specify the Email Addresses on a Distribution List	575
ps_email_address_expanded Attribute—Specify Additional Email Addresses on a Distribution List	576
ps_email_log Attribute—Specify Whether to Email PeopleSoft Job Logs	577
ps_email_subject Attribute—Define an Email Subject	578
ps_email_text Attribute—Define the Body Text of an Email	579
ps_email_web_report Attribute—Specify Whether to Email a PeopleSoft Web Report	580
ps_operator_id Attribute—Specify a PeopleSoft Operator ID	581
ps_output_dest Attribute—Specify an Output Path for a PeopleSoft Job	582
ps_process_name Attribute—Specify a PeopleSoft Process to Run	585
ps_process_type Attribute—Specify the Type of PeopleSoft Process to Run	586
ps_restarts Attribute—Specify Whether to Disable the Restart of Failed PeopleSoft Jobs	587
ps_run_cntrl_args Attribute—Specify the Arguments for a PeopleSoft Run Control Table	588
ps_run_cntrl_id Attribute—Specify PeopleSoft Run Parameters	591
ps_run_control_table Attribute—Specify the Table that Contains the PeopleSoft Run Parameters	592
ps_server_name Attribute—Specify a Server to Run the PeopleSoft Job	594
ps_skip_parm_updates Attribute—Specify Whether to Update Job Parameters	595
ps_time_zone Attribute—Specify a Time Zone	597
remote_name Attribute—Specify a Remote Class Name	598
request_id Attribute—Specify Request ID of the Oracle E-Business Suite Request to Copy	599
result_type Attribute—Specify the Data Type to be Returned from the Stored Procedure	600
return_class_name Attribute—Specify a Return Class Name	601
return_namespace Attribute—Specify an XML Namespace	602
return_xml_name Attribute—Specify an XML Type	604
run_calendar Attribute—Identify a Custom Calendar	606
run_window Attribute—Define an Interval for a Job to Start	607
sap_abap_name Attribute—Run ABAP Steps Within an SAP Job	609
sap_chain_id Attribute—Identify Name of Business Warehouse Process Chain on SAP	610

sap_client Attribute—Specify SAP Client Number	611
sap_event_id—Specify the SAP Event to Monitor or Trigger	612
sap_event_parm—Specify an SAP Event Parameter	613
sap_ext_table Attribute—Specify Data Selection Criteria	614
sap_fail_msg Attribute—Specify Failure Message for SAP Job	615
sap_info_pack Attribute—Identify Name of Business Warehouse InfoPackage on SAP System	616
sap_is_trigger Attribute—Specify the Action to Take on an SAP Event	617
sap_job_class Attribute—Specify SAP Job Class	618
sap_job_count Attribute—Specify the Job ID of the SAP Job to Copy	619
sap_job_name Attribute—Specify SAP Job Name	620
sap_lang Attribute—Specify SAP Language for Login	622
sap_mon_child Attribute—Enable Monitoring of Children Jobs	623
sap_office Attribute—Save Outgoing Documents to SAPoffice Outbox	624
sap_print_parms Attribute—Specify Print Parameters	625
sap_process_client Attribute—Specify the Client to Monitor for a Specified Process	629
sap_process_status Attribute—Specify the SAP Process Status to Monitor	630
sap_proc_type Attribute—Specify the SAP Process Type to Monitor	631
sap_proc_user—Specify an SAP Process User	632
sap_recipients Attribute—Specify SAP Spool Recipient	633
sap_release_option Attribute—Specify Release Option for a Job	635
sap_rfc_dest Attribute—Specify SAP Destination	636
sap_step_num Attribute—Specify Number of First Step to Copy	637
sap_step_parms Attribute—Specify SAP R/3 Step Specifications	638
sap_success_msg Attribute—Specify Success Message for SAP Job	647
sap_target_jobname Attribute—Specify Job Name to Copy	648
sap_target_sys Attribute—Specify SAP Application Server	649
scp_local_name Attribute—Specify the Local File to Transfer	650
scp_local_user Attribute—Specify a User ID on the Agent Computer	652
scp_protocol Attribute—Specify the SCP Transfer Protocol to Use	653
scp_remote_dir Attribute—Specify a Remote Directory	654
scp_remote_name Attribute—Specify a Remote File Name	656
scp_server_name Attribute—Specify a Remote Server Name	658
scp_server_port Attribute—Specify a Remote Server Port	659
scp_target_os Attribute—Specify the Remote Operating System Type	660
scp_transfer_direction Attribute—Specify the File Transfer Direction	661
search_bw Attribute—Define the Time Used to Search Backwards for a z/OS Manual Job	662
send_notification Attribute—Specify Whether to Send a Notification When a Job Completes	664
service_desk Attribute—Specify Whether to Open a CA Service Desk Ticket When a Job Fails	665
service_name Attribute—Specify the Web Service Name Within the Target Namespace	666
shell Attribute—Specify a UNIX Shell to Run a Script or Command	668
sp_arg Attribute—Specify a Parameter to Pass to a Stored Procedure	670
Supported Data Types	672

Handling Unsupported Data Types.....	673
sp_name Attribute—Specify Stored Procedure or Stored Function to Run	673
sql_command Attribute—Specify an SQL Statement to Run.....	674
start_mins Attribute—Define the Minutes to Run a Job	676
start_times Attribute—Define the Time of the Day to Run a Job	677
std_err_file Attribute—Redirect the Standard Error File	679
std_in_file Attribute—Redirect the Standard Input File	683
std_out_file Attribute—Redirect the Standard Output File.....	687
success_codes Attribute—Define Exit Codes to Indicate Job Success	692
success_criteria Attribute—Specify the Evaluation Criteria for a Return String	693
success_pattern Attribute—Specify a Success Pattern Using Regular Expression Logic	696
svcdesk_attr Attribute—Define CA Service Desk Request Attributes	698
svcdesk_desc Attribute—Define the Message to Include in the CA Service Desk Request	700
svcdesk_imp Attribute—Specify the Impact Level of the CA Service Desk Request	701
svcdesk_pri Attribute—Specify the Priority Level of the CA Service Desk Request	702
svcdesk_sev Attribute—Specify the Severity Level of the CA Service Desk Request.....	703
tablename Attribute—Specify the Database Table to Monitor.....	704
target_namespace Attribute—Specify a Target Namespace.....	705
term_run_time Attribute—Specify the Maximum Runtime	706
text_file_filter Attribute—Specify a Text String to Search For	707
text_file_filter_exists Attribute—Specify Whether to Monitor for the Existence of Text.....	710
text_file_mode Attribute—Specify the Search Mode	711
text_file_name Attribute—Specify a Text File Name and Location	715
time_format Attribute—Define a Time Format	717
time_position Attribute—Specify the First Column of a Time Stamp	720
timezone Attribute—Define the Time Zone	722
trigger_cond Attribute—Specify a Condition to Monitor For	724
trigger_type Attribute—Specify the Type of Database Change to Monitor For.....	726
ulimit Attribute—Specify UNIX Resource Limits	729
upper_boundary Attribute—Define the End of the Range to be Searched	731
upper_boundary Attribute—Define the Maximum CPU or Disk Usage to Monitor (OMCPU and OMD Jobs) ...	734
URL Attribute—Specify the URL of the JMX Server	736
use_topic Attribute—Publish or Subscribe to a Topic or Queue	737
user_role Attribute—Specify Database Resource Location	738
watch_file Attribute—Specify a File to Monitor	740
File Trigger Notes	741
Examples: Specifying UNIX files using the watch_file Attribute	743
Examples: Specifying Windows files using the watch_file Attribute	743
Examples: Specifying i5/OS files using the watch_file Attribute	744
watch_file_change_type Attribute—Specify the Type of Change in File Size	746
watch_file_change_value Attribute—Specify the Change in File Size	748
watch_file_groupname Attribute—Specify the Group that Owns the File	750

watch_file_min_size Attribute—Specify the Minimum Size that a File Must Reach	751
watch_file_owner Attribute—Specify the Owner of the File to be Monitored	752
watch_file_recursive Attribute—Specify Whether to Monitor Subdirectories for File Activity	754
watch_file_type Attribute—Specify the Type of File Activity to Monitor For	756
CREATE File Trigger Type Notes	757
DELETE File Trigger Type Notes	758
EXIST File Trigger Type Notes	758
EXPAND File Trigger Type Notes	759
NOTEXIST File Trigger Type Notes	762
SHRINK File Trigger Type Notes	763
UPDATE File Trigger Type Notes	767
GENERATE File Trigger Type Notes	768
watch_file_win_user Attribute—Specify a Windows User for Monitoring UNC Files	769
watch_interval Attribute—Specify the Frequency to Monitor a File	770
watch_no_change Attribute—Specify the Time the File Must Remain Unchanged	772
web_parameter Attribute—Specify Operation Parameters	773
web_user Attribute—Specify a Web Service User Name	776
win_event_category Attribute—Specify a Windows Event Category	777
win_event_computer Attribute—Specify a Local Computer for Windows Event Log	778
win_event_datetime Attribute—Specify the Date and Time of a Windows Event Log	779
win_event_description Attribute—Specify a Windows Event Description	780
win_event_id Attribute—Specify a Windows Event ID	781
win_event_op Attribute—Specify a Comparison Operator for a Windows Event ID	782
win_event_source Attribute—Specify a Windows Event Source	783
win_event_type Attribute—Specify a Windows Event Type	784
win_log_name Attribute—Specify the Name of a Windows Event Log	785
win_service_name Attribute—Specify the Name of the Windows Service to be Monitored	786
win_service_status Attribute—Specify the Status of the Windows Service to be Monitored	787
wsdl_operation Attribute—Specify the Operation to be Invoked	788
WSDL_URL Attribute—Specify a WSDL URL	790
zos_dataset Attribute—Specify a JCL Library for a z/OS Data Set Trigger Job	792
zos_dsn_renamed Attribute—Specify Whether to Monitor When a Data Set is Renamed	793
zos_dsn_updated Attribute—Specify Whether to Monitor for Updates to a Data Set	794
zos_explicit_dsn Attribute—Specify Whether to Monitor for an Explicit Data Set Notification	795
zos_ftp_direction Attribute—Specify Whether to Monitor for an FTP Transfer To or From a Remote Computer	796
zos_ftp_host Attribute—Specify a Remote Computer for an FTP Transfer	797
zos_ftp_userid Attribute—Specify a User ID for an FTP Connection	798
zos_jobname Attribute—Specify the z/OS Job Name	799
zos_trigger_by Attribute—Specify a Job Name or a User ID that Triggers the Job	800
zos_trigger_on Attribute—Specify the Number of Actions that Must Occur	802
zos_trigger_type Attribute—Specify Whether to Monitor for Data Set Activity by a Job or a User ID	803

Chapter 5: JIL User-Defined Job Type Definitions 805

delete_job_type Subcommand—Delete a Job Type from the Database	805
insert_job_type Subcommand—Add a Job Type to the Database	806
update_job_type Subcommand—Update a Job Type in the Database	807
command Attribute—Specify Command for the Job Type	807
description Attribute—Specify a Description for the Job Type	808

Chapter 6: JIL Machine Definitions 809

delete_machine Subcommand—Delete a Real or Virtual Machine	809
Delete a Real Machine	810
Delete a Virtual Machine	811
Delete a Real Machine Pool	811
Delete a Reference to a Real Machine	811
Delete All References to a Real Machine and the Machine Itself	812
insert_machine Subcommand—Add a Machine Definition	813
Real Machines	817
Virtual Machines	818
Real Machine Pools	819
update_machine Subcommand—Update a Machine Definition	820
Updating Virtual Machines	821
agent_name Attribute—Specify the Name of an Agent	821
character_code Attribute—Specify the Character Code for the Machine	823
description Attribute—Specify a Description for an Agent	824
encryption_type Attribute—Specify the Type of Encryption	824
factor Attribute—Define a Factor for an Agent Machine	826
force Attribute—Force a Machine to be Deleted	828
key_to_agent Attribute—Specify the Agent Encryption Key	829
machine Attribute—Define or Update a Real Machine As a Component of the Virtual Machine or Real Machine Pool	830
max_load Attribute—Define the Maximum Load an Agent Machine Can Handle	831
node_name Attribute—Specify the IP Address of a Machine	833
opsys Attribute—Specify the Operating System Type of a Machine	833
port Attribute—Specify the Port Used to Communicate with the Machine	835
remove_references Attribute—Remove All Machine References from Virtual Machines or Real Machine Pools	836
type Attribute—Specify the Machine Type	837

Chapter 7: JIL Resource Definitions 839

delete_resource Subcommand—Delete a Virtual Resource	839
insert_resource Subcommand—Define a Virtual Resource	840

update_resource Subcommand—Update a Virtual Resource	842
amount Attribute—Specify the Amount of Virtual Resource	844
description Attribute—Specify a Description for a Virtual Resource	845
machine Attribute—Specify the Machine for a Virtual Resource	846
res_type Attribute—Specify the Type of Virtual Resource	847
resources Attribute—Define or Update Real and Virtual Resource Dependencies in a Job	849
Considerations When Specifying the real_resource_type Option	853

Chapter 8: JIL Cross-Instance Definitions 855

delete_xinst Subcommand—Delete an External Instance	855
insert_xinst Subcommand—Define an External Instance	856
update_xinst Subcommand—Update an External Instance Definition	858
xcrypt_type Attribute—Specify the Encryption Type	860
xkey_to_manager Attribute—Specify an Encryption Key for Manager Communication	861
xmachine Attribute—Define Connection Information for an External Instance	862
xmanager Attribute—Specify the Alias Name of a CA Workload Automation EE Instance	863
xport Attribute—Specify the Port Number of the External Instance	864
xtype Attribute—Specify the External Instance Type	865

Chapter 9: JIL Monitor and Report Definitions 867

delete_monbro Subcommand—Delete a Monitor or Report	867
insert_monbro Subcommand—Define a Monitor or Report	868
update_monbro Subcommand—Update a Monitor or Report	870
after_time Attribute—Specify the Report Start Date and Time	871
alarm Attribute—Specify Whether to Track Alarms	873
alarm_verif Attribute—Specify Whether the Monitor Requires an Operator Response to Alarms	874
all_events Attribute—Specify Whether to Track all Events for a Job	875
all_status Attribute—Specify Whether to Track all Job Status Events	876
currun Attribute—Specify Whether to Report Events in the Current Run	877
failure Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to FAILURE	878
job_filter Attribute—Specify the Jobs to Track	879
job_name Attribute—Identify the Job for Which to Track Events	880
mode Attribute—Specify Whether to Define a Monitor or Report	881
restart Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to RESTART	882
running Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to RUNNING	883
starting Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to STARTING	884
success Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to SUCCESS	885
terminated Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to TERMINATED ..	886

Chapter 10: JIL Blob and Glob Definitions 887

delete_blob Subcommand—Delete a Blob from the Database	887
delete_glob Subcommand—Delete a Glob from the Database	888
insert_blob Subcommand—Add a Blob to the Database	888
insert_glob Subcommand—Add a Glob to the Database	890
blob_file Attribute—Specify File for the Blob	891
blob_input Attribute—Specify Input for the Blob	891
blob mode Attribute—Specify File Format for the Glob or Blob	892
blob_type Attribute—Specify File Type of the Job Blob	893

Chapter 11: System States 895

Events	895
ALARM (106)	896
ALERT (139)	897
CHANGE_PRIORITY (120)	897
CHANGE_STATUS (101)	897
CHECK_HEARTBEAT (116)	898
CHK_BOX_TERM (118)	898
CHK_MAX_ALARM (114)	898
CHK_RUN_WINDOW (122)	898
CHK_COMPLETE (136)	898
CHK_START (135)	898
CHK_TERM_RUNTIME (138)	899
COMMENT (117)	899
DELETEJOB (119)	899
EXTERNAL_DEPENDENCY (127)	899
FORCE_STARTJOB (108)	900
HEARTBEAT (115)	900
INVALIDATE_MACH (133)	900
JOB_OFF_HOLD (113)	900
JOB_OFF_ICE (111)	901
JOB_ON_HOLD (112)	901
JOB_ON_ICE (110)	901
KILLJOB (105)	902
MACH_OFFLINE (130)	903
MACH_ONLINE (131)	903
MACH_PROVISION (142)	903
REFRESH_EXTINST (129)	904
RELEASE_RESOURCE (137)	904
REPLY_RESPONSE (140)	904
RESTARTJOB (143)	904

SET_GLOBAL (125)	904
SEND_SIGNAL (126)	905
STARTJOB (107)	905
STATE_CHANGE (141)	905
STOP_DEMON (109)	905
Status	906
ACTIVATED (9)	906
FAILURE (5)	907
INACTIVE (8)	907
PEND_MACH (14)	907
ON_HOLD (11)	907
ON_ICE (7)	907
QUE_WAIT (12)	908
RESTART (10)	908
RESWAIT (15)	908
RUNNING (1)	908
STARTING (3)	908
SUCCESS (4)	908
TERMINATED (6)	909
WAIT_REPLY (13)	909
Alarms	909
AUTO_PING (526)	910
CHASE (514)	910
CPI_JOBNAME_INVALID (535)	911
CPI_UNAVAILABLE (536)	911
DATABASE_COMM (516)	911
DB_PROBLEM (523)	911
DB_ROLLOVER (519)	911
DUPLICATE_EVENT (524)	911
EP_HIGH_AVAIL (522)	912
EP_ROLLOVER (520)	912
EP_SHUTDOWN (521)	912
EVENT_HDLR_ERROR (507)	912
EVENT_QUE_ERROR (508)	912
EXTERN_DEPS_ERROR (529)	912
FORKFAIL (501)	913
KILLJOBFAIL (540)	913
INSTANCE_UNAVAILABLE (525)	913
JOBFAILURE (503)	913
JOBNOT_ONICEHOLD (509)	913
MACHINE_UNAVAILABLE (532)	913
MAX_RETRY (505)	913

MAXRUNALARM (510)	914
MINRUNALARM (502)	914
MISSING_HEARTBEAT (513)	914
MUST_COMPLETE_ALARM (538)	914
MUST_START_ALARM (537)	914
REPLY_RESPONSE_FAIL (542)	914
RESOURCE (512)	915
RESTARTJOBFAIL (544)	915
RETURN_RESOURCE_FAIL (543)	915
SENDSIGFAIL (541)	915
SERVICEDESK_FAILURE (533)	915
STARTJOBFAIL (506)	915
UNINOTIFY_FAILURE (534)	916
VERSION_MISMATCH (518)	916
WAIT_REPLY_ALARM (539)	916
Exit Codes	917

Chapter 12: Database Tables and Views 919

ujointerface_afm	919
ujointerface_afm_strings	920
ujointerface_agent_alias	920
ujointerface_alamode	920
ujointerface_alarm	921
ujointerface_asbnode	921
ujointerface_asext_config	922
ujointerface_audit_info	922
ujointerface_audit_msg	923
ujointerface_avg_job_runs	923
ujointerface_calendar	923
ujointerface_chase	924
ujointerface_chkpkt_rstart	924
ujointerface_comm_recv_seq	925
ujointerface_comm_send_seq	925
ujointerface_command_job	926
ujointerface_cred	926
ujointerface_cycle	927
ujointerface_event	927
ujointerface_event2	929
ujointerface_eventvu	929
ujointerface_ext_calendar	930
ujointerface_ext_event	930

uko_ext_job	932
uko_extended_jobrun_info	932
uko_file_watch_job	933
uko_ftp_job	933
uko_glob	934
uko_globblob	934
uko_ha_process	934
uko_i5_job	935
uko_intcodes	935
uko_job	936
uko_job_cond	937
uko_job_resource_dep	938
uko_job_runs	938
uko_job_status	939
uko_job_tree	939
uko_jobblob	940
uko_jobst	941
uko_jobtype	942
uko_keymaster	942
uko_last_eoid_counter	942
uko_ma_process	943
uko_machine	943
uko_meta_enumerations	944
uko_meta_properties	944
uko_meta_rules	945
uko_meta_types	945
uko_micro_focus_job	946
uko_monbro	946
uko_monitor_object_job	947
uko_monitor_winevent_job	947
uko_msg_ack	948
uko_next_oid	948
uko_oraaps	949
uko_oraaps_steps	949
uko_patch	950
uko_peoplesoft_job	950
uko_proc_event	951
uko_real_resource	952
uko_rep_daily	953
uko_rep_hourly	954
uko_rep_monthly	955
uko_rep_weekly	956

uko_req_job	957
uko_reswait_que	957
uko_sap_arcspec	958
uko_sap_infodetail	958
uko_sap_job	959
uko_sap_jobstep	959
uko_sap_prtspec	960
uko_sap_recipient	961
uko_sched_info	962
uko_service_desk	962
uko_snmp_job	963
uko_sql_job	963
uko_sql_proc_parms	964
uko_strings	964
uko_sys_ha_state	964
uko_temp_jobruns	964
uko_timezones	965
uko_uninotify	965
uko_virt_resource	965
uko_virt_resource_lookup	966
uko_virt_resource_status	966
uko_wait_que	967
uko_web_services	968
uko_web_services2	969
uko_wol_job	969
uko_workflow	970
uko_zos_condcodes	970
uko_zos_dsn_trigger	971
uko_zos_job	971

Index

973

Chapter 1: Introduction

CA Workload Automation AE centralizes and automates the scheduling and management of jobs in distributed UNIX and Windows environments.

Intended Audience

This document is for users who are responsible for defining, scheduling, monitoring, or controlling workload. This document describes detailed syntax for commands and job, machine, monitor, and report definition parameters. This document also lists valid system states.

To use this document, you must be familiar with CA Workload Automation AE, the operating system(s) where the jobs run and any third-party products or software technology that the jobs use.

Notes:

- The UNIX instructions in this document also apply to Linux systems unless otherwise noted.
- The term *Windows* refers to any Microsoft Windows operating system supported by CA Workload Automation AE unless otherwise noted.
- This document is a supplement to the *User Guide*. For more information about concepts and procedures, see the *User Guide*.

Legacy Agent Replaced by CA Workload Automation Agent

The new CA Workload Automation Agent for UNIX, Linux, or Windows replaces the Remote Agent (`auto_remote`) that was provided with Unicenter AutoSys JM r4.5 and r11. The r11.3 documentation refers to `auto_remote` as the *legacy agent*.

The new agent provides additional job types, including monitoring and FTP jobs. The agent is automatically installed on the computer where CA Workload Automation AE is installed. You can also install the agent on remote computers to run jobs on those computers.

Chapter 2: Commands

This chapter provides an alphabetical listing of the commands used to control, configure, and report on jobs in CA Workload Automation AE.

JIL Syntax Rules

JIL scripts contain one or more JIL subcommands and one or more attribute statements; these elements constitute a job definition. When writing a JIL script, you must follow these syntax rules:

Rule 1

Each subcommand uses the following form:

sub_command:object_name

sub_command

Defines a JIL subcommand.

object_name

Defines the name of the object (for example, a job or a machine) to act on.

Rule 2

You can follow each subcommand with one or more attribute statements. These statements can occur in any order and are applied to the object specified in the preceding subcommand. A subsequent subcommand begins a new set of attributes for a different object. The attribute statements have the following form:

attribute_keyword:value

attribute_keyword

Defines a valid JIL attribute.

value

Defines the setting to apply to the attribute.

Notes:

- The entire *attribute_keyword:value* statement can be up to 4096 characters.
- The following subcommands do not accept attributes: `delete_box`, `delete_job`, `delete_job_type`, `delete_xinst`, `delete_monbro`, and `delete_glob`.

Rule 3

You can enter multiple attribute statements on the same line, but you must separate the attribute statements with at least one space.

Rule 4

A box job definition must exist before you can add jobs to it.

Rule 5

Valid *value* settings can include any of the following characters:

- Uppercase and lowercase letters (A-Z, a-z)
- Hyphens (-)
- Underscores (_)
- Pound signs (#)
- Numbers (0-9)
- Colons (:), if the colon is escaped with quotation marks (" ") or a preceding backslash (\)
- The at character (@)

Note: Object names can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (_), hyphen (-), and pound (#). Do not include embedded spaces or tabs.

Rule 6

Because JIL parses on the combination of a keyword followed by a colon, you must use escape characters (a backslash) or enclose the value in quotation marks with any colons used in an attribute statement's value. For example, to define the start time for a job, specify 10\:00 or "10:00".

Note: When specifying drive letters in commands, you must use escape characters with the colon (:). For example, "C:\tmp" and C\:tmp are valid; C:\tmp is not.

Rule 7

Use one of the following methods to indicate comments in a JIL script:

- To comment an entire line, put a pound sign (#) in the first column.
- To comment one or more lines, you can use the C programming syntax for beginning a comment with a forward slash and asterisk /*) and ending it with an asterisk and a forward slash (*/). For example:

```
/* this is a comment */
```

Rule 8

You can use the blob_input attribute to manually enter multiline text. This attribute is only valid for the insert_job, update_job, insert_blob, and insert_glob subcommands. The blob_input attribute has the following form:

```
blob_input:<auto_blob> this is a multi-
line text
</auto_blob>
```

Note: Use the auto_blob meta-tags to indicate the beginning and end of multiline text. JIL interprets every character input between the auto_blob meta-tags literally. This implies that JIL does not enforce any of the previously discussed rules for text entered in an open auto_blob meta-tag.

Rule 9

To specify a comma in *keyword=value* combinations, you must do *one* of the following:

- Escape the comma with a backslash, as follows:

```
j2ee_parameter: String=Hello1\, World
```

- Enclose the entire value in quotation marks, as follows:

```
j2ee_parameter: String="Hello1, World"
```

If the *keyword=value* combination contains commas and quotation marks, you must escape the commas and quotation marks as follows:

```
j2ee_parameter: String=\"Hello1\, World\"
```

Issuing Commands on UNIX

You can run commands (including appropriate parameters) at the UNIX operating system prompt, after the CA Workload Automation AE environment is sourced. You can also embed many of the commands discussed in this chapter in shell scripts and create aliases for commands you use frequently.

The command reference in this chapter is also available online, through the UNIX man command. For example, to access the reference page for the sendevent command, enter:

```
man sendevent
```

If this action does not display the proper man page, it is because the MANPATH environment variable is not set correctly. This variable is usually set in the \$AUTOUSER/autosys.* files used for setting up the environment (where * denotes the .csh.host name, .sh.host name, or ksh.host name).

Issuing Commands on Windows

You must use the CA Workload Automation AE Instance Command Prompt (located in the CA Workload Automation AE program group) to execute commands. This command prompt presets all of the environment variables for the instance. Commands will not run if executed from an MS-DOS command prompt.

You can also embed many of the commands discussed in this chapter in batch files. You run those batch files from the CA Workload Automation AE Instance Command Prompt.

When you use issue commands that run on a different operating system, you must use the syntax appropriate to the operating system where the job runs.

Wildcard Characters in Commands

Some commands let you specify wildcard characters to select one or more items as part of a command. The percent (%) and asterisk (*) characters can be used to represent any number of generic characters. The question mark (?) character can be used to represent any single generic character.

For example, specifying -J %box% in the autorep command generates reports for all items whose names contain the string "box". Whereas, specifying -J a? in the autorep command generates reports for all items whose names are two characters in length beginning with "a".

Note: When on UNIX or Linux platforms, the "*" and "?" characters may be substituted by the command shell with matching file or directory names. To pass these characters, as is, to programs, they must be enclosed in quotation marks. For example:

```
autorep -q -J "a*"
```

See the specific command and parameter descriptions in this chapter for information about when you can use wildcard characters.

Commands by Task

The following table shows commands used for specific tasks. Unless otherwise specified, all commands are for both UNIX and Windows.

Task	Command
Check system status	autoflags autoping autosyslog chase chk_auto_up
Convert cron to jil (UNIX only)	cron2jil
Define objects (for example, jobs and machines)	jil
Define calendars	autocal_asc
Event commands	sendevent
Maintain databases	archive_events archive_jobs autotrack clean_files dbspace dbstatistics DBMaint
Manage security	autosys_secure
Manage time zone tables	autotimezone
Monitor jobs	autorep autosyslog monbro
Report job dependencies and conditions	job_depends
Report job status	autorep autostatad autostatus forecast monbro
Start CA Workload Automation AE (UNIX Only)	eventor

archive_events Command—Archive Data

The archive_events command is a client and scheduler component utility that archives data from the CA Workload Automation AE database. It removes data that is older than the specified number of days from the CA Workload Automation AE ujo_proc_event, ujo_job_runs, ujo_audit_info, ujo_chkpkt_rstart, ujo_audit_msg tables, ujo_alarm, and ujo_extended_jobrun_info and archives it to a flat file. Use archive_events to prevent the database from becoming full.

In dual event server mode, the data is archived from both the event servers at the same time. If information from these tables is not regularly purged from the database, the database can fill up, stopping all processing. We recommend that you run archive_events during the database maintenance cycle.

The DBMaint script, which runs daily by default, runs archive_events as a part of its process.

Syntax

This command has the following format:

```
archive_events [-A]
               [-B batch_size]
               [-C delimiter]
               [-d directory_name]
               [-D data_server:database|TNSname]
               [-j num_days|-l num_days|-n num_days|-r num_days]
               [-x]
               [-?]
```

-A

(Optional) Deletes the information after archiving it to a file. If you omit the -A argument, the information is deleted without archiving.

-B *batch_size*

(Optional) Defines the number of events to archive at a time.

Limits: 1-50000

Default: 1000

Notes:

- Use the default value unless the database is full and the transaction log is likely to become full if 1000 events are archived at once. Because the database transaction file records each transaction (in this case, the deletion of a single event), and the transaction file shares file space with the actual data, only archive a few events at a time when the amount of available file space is low. If there is ample space in the transaction log file, a larger value can be specified to reduce the archival time.
- If you do not maintain low or optimum job load, the transaction log fills up and the transactions are suspended. In dual event server mode, CA Workload Automation AE rolls over to single event server mode if the database transaction log on the primary event server is full.

-C *delimiter*

(Optional) Defines a character to use as a delimiter for columns in output archive files.

Default: Comma (,)

Limits: This value must be one character.

-d *directory_name*

(Optional) Defines the directory in which to store archived data. If you omit this parameter, the product archives data to the default directory (\$AUTOUSER/archive for UNIX, or %AUTOUSER%\archive for Windows).

-D *data_server:database/TNSname*

(Optional) Specifies the data source from which to archive events.

data_server:database

Specifies the Sybase data server and database from which to archive events.

TNSname

Specifies the TNS alias of the Oracle data server from which to archive events.

-j *num_days*

(Optional) Deletes job runs information that is older than the *num_days* value from the ujo_job_runs table. The *num_days* value indicates the number of days to leave information about job runs in the database. When a row in the table exceeds this value, the row is deleted.

-l *num_days*

(Optional) Deletes audit information older than the *num_days* value from the ujo_audit_info and ujo_audit_msg tables. The *num_days* value indicates the number of days to leave information about audits in the database. When a row in the table exceeds this value, the row is deleted.

-n *num_days*

(Optional) Deletes events and alarms information older than the *num_days* value from the ujo_proc_event and ujo_alarms table. The *num_days* value indicates the number of days to leave events and their associated alarms in the database. When a row in the table exceeds this value, the row is deleted.

-r *num_days*

(Optional) Deletes checkpoint information older than the *num_days* value from the ujo_chkpt_rstart table. The *num_days* value indicates the number of days to leave information about checkpoints in the database. When a row in the table exceeds this value, the row is deleted.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Copy Events Older than Five Days

This example copies all events older than five days from the ujo_proc_event table to the default archive file and deletes them from the database.

```
archive_events -A -n 5
```

Example: Copy All job_runs Statistics Older than Five Days

This example copies all job_runs statistics older than five days to a specified archive directory and deletes them from the database.

```
archive_events -A -d \my_archive -j 5
```

More information:

[ujointerface_audit_info](#) (see page 922)
[ujointerface_audit_msg](#) (see page 923)
[ujointerface_job_runs](#) (see page 938)
[ujointerface_proc_event](#) (see page 951)
[ujointerface_chkpt_rstart](#) (see page 924)
[ujointerface_alarm](#) (see page 921)

Archive Files

The archive_events command removes data from the following database tables and stores the data in the corresponding flat files:

ujointerface_audit_info

The archive_events command archives data from the ujo_audit_info table to the following file:

- archived_audit.\$AUTOSERV.MM.dd.yyyy.hh.mm.ss (UNIX)
- archived_audit.%AUTOSERV%.MM.dd.yyyy.hh.mm.ss (Windows)

MM.dd.yyyy.hh.mm.ss

Indicates the month, day, year, hour, minute, and second when the archive was created.

The format of each record in the flat file is as follows:

audit_info_num,entity,time,type

For example:

```
4324,user@domain,1104868885,J
```

Each audit event occupies one row in the archive. By default, a comma (,) delimits each column.

ujointerface_audit_msg

The archive_events command archives data from the ujo_audit_msg table to the following file:

- archived_audit_msg.\$AUTOSERV.MM.dd.yyyy.hh.mm.ss (UNIX)
- archived_audit_msg.%AUTOSERV%.MM.dd.yyyy.hh.mm.ss (Windows)

MM.dd.yyyy.hh.mm.ss

Indicates the month, day, year, hour, minute, and second when the archive was created.

The format of each record in the flat file is as follows:

audit_info_num,seq_no,attribute,value,value2,is_edit

For example:

4315,1,job_name,xxforcestart_jobs_inbox_test6,,N

Each event occupies one row in the archive. By default, a comma (,) delimits each column.

ujointerface_job_runs

The archive_events command archives data from the ujo_job_runs table to the following file:

- archived_job_runs.\$AUTOSERV.MM.dd.yyyy.hh.mm.ss (UNIX)
- archived_job_runs.%AUTOSERV%.MM.dd.yyyy.hh.mm.ss (Windows)

MM.dd.yyyy.hh.mm.ss

Indicates the month, day, year, hour, minute, and second when the archive was created.

The format of each record in the flat file is as follows:

joid,run_num,ntry,starttime,endtime,status,exit_code,runtime,evt_num,machine

For example:

11679,2508,1,1104442415,1104442417,4,0,2,33366,mach3

Each job run occupies one row in the archive. By default, a comma (,) delimits each column.

ujointerfaceproc_event

The archive_events command archives data from the ujo_proc_event table to the following file:

- archived_events.\$AUTOSERV.MM.dd.yyyy.hh.mm.ss (UNIX)
- archived_events.%AUTOSERV%.MM.dd.yyyy.hh.mm.ss (Windows)

MM.dd.yyyy.hh.mm.ss

Indicates the month, day, year, hour, minute, and second when the archive was created.

The format of each record in the flat file is as follows:

```
eoid,joid,job_name,box_name,AUTOSERV,priority,event,status,alarm,event_time_g  
mt,exit_code,machine,pid,jc_pid,run_num,ntry,text,que_priority,stamp,evt_num,  
que_status,que_status_stamp
```

For example:

```
ING00000oi100,11051,box_alarms_sleeper2,box_alarms_max_term,ING,0,106,0,507,11  
04436975,0,localhost,0,0,2365,1,"Cannot TERMINATE Job: box_alarms_sleeper2  
because its in the STARTING state- Try later",0,30-dec-2004  
15:02:55,1,2,30-dec-2004 15:02:55
```

Each event occupies one row in the archive. By default, a comma (,) delimits each column.

ujointerfacechkpnt_rstart

The archive_events command archives data from the ujo_chkpkt_rstart table to the following file:

- archived_chkpkt_rstart.\$AUTOSERV.MM.dd.yyyy.hh.mm.ss (UNIX)
- archived_chkpkt_rstart.%AUTOSERV%.MM.dd.yyyy.hh.mm.ss (Windows)

MM.dd.yyyy.hh.mm.ss

Indicates the month, day, year, hour, minute, and second when the archive was created.

The format of each record in the flat file is as follows:

```
dest_machine,dest_app,as_evt_time,ubc_name,ubc_jobnumbr,ubc_sysid,ubc_date,ub  
c_time,ubc_procid,ubc_userid,ubc_compcodes,ubc_jobname,ubc_setname,ubc_jobnumb  
,ubc_server,ubc_from_sysid,enefill,ubt_cputime,ubt_errcod
```

Each record occupies one row in the archive. By default, a comma (,) delimits each column.

More information:

[ujc_chkpt_rstart](#) (see page 924)
[ujc_job_runs](#) (see page 938)
[ujc_proc_event](#) (see page 951)
[ujc_audit_info](#) (see page 922)
[ujc_audit_msg](#) (see page 923)

archive_jobs Command—Delete Obsolete Job Versions

The archive_jobs command deletes obsolete job versions from the database. Job versions are obsolete when the job is inactive and the database no longer refers to it. The archive_jobs command can help you prevent the database from being overloaded with obsolete job versions. We recommend that you issue the archive_events command before issuing the archive_jobs command. We also recommend that you run archive_jobs as part of your usual database maintenance.

Syntax

This command has the following format:

`archive_jobs -j number_of_days [-d "directory_name"] [-A] [-x] [-?]`

-j *number_of_days*

Specifies a number of days. Obsolete jobs older than this number are deleted.

-d "directory_name"

(Optional) Specifies the name of the directory where the archive file is created.

-A

(Optional) Copies the job definitions to an archive file.

Default: \$AUTOUSER/archive

-x

(Optional) Displays the version number.

-?

(Optional) Displays help for the command.

Example: Delete Obsolete Job Versions

- This example deletes obsolete job versions older than 7 days:
`archive_jobs -j 7`
- This example deletes obsolete job versions older than 7 days and creates the archive flat file in the \$AUTOUSER/archive file (the default):
`archive_jobs -j 7 -A`
- This example deletes obsolete job versions older than 7 days and creates the archive flat file in the /tmp/archive directory:

```
archive_jobs -j 7 -A -d "/tmp/archive"
```

as_config Command—Manage Encryption Keys and Passwords

The as_config command lets you manage the encryption keys for the application server and for CA Workload Automation Agent for UNIX, Linux, or Windows. You can use as_config to encrypt a key and store it in \$AUTOUSER/cryptkey.txt. If you want to use that encrypted key for the agent, you can copy the cryptkey.txt file to the *install_directory/SystemAgent/agent_name* directory.

Syntax

This command has the following format:

```
as_config [-g key] [-a key] [-d] [ -p | -p -n pwvar ] [ -m message_key | -m message_key argument... ] [-x] [-?]
```

-g key

(Optional) Encrypts the specified key and stores it in \$AUTOUSER/cryptkey.txt.

Note: To use the -g option, you must set the AUTOUSER environment variable.

-a key

(Optional) Encrypts the specified key and stores it in \$AUTOUSER/cryptkey_alias.txt.
This option supports 32-character hexadecimal key only.

Note: To use the -a option, you must set the AUTOUSER environment variable.

-d

Encrypts the default encryption key and stores it in \$AUTOUSER/cryptkey.txt.

-p

(Optional) Decodes the passwords and sends it to the standard output.

-n pwvar

(Optional) Defines an environment variable containing the password. The pwvar variable defines the password for the CA EEM server.

Default: If you do not define this parameter, the password is read from standard input.

-m message_key

(Optional) Displays the message that corresponds to the specified key.

Note: To use the -m option, you must set the AUTOSYS environment variable.

Example: as_config -m CUAJM_I_40005

argument ...

Specifies one or more arguments to pass to the -m option.

-x

(Optional) Displays the version information.

-?

(Optional) Displays help for the command.

as_info Command—Return Installation Information

Valid on UNIX

The as_info command is a client component utility that returns CA Workload Automation AE installation information to standard output.

When you issue as_info with no parameters, the command returns the CA Workload Automation AE components and version installed, installation log locations, configured instances, and status.

Syntax

This command has the following format:

as_info [-a] [-c] [-d] [-f] [-i] [-l] [-p] [-s] [-v]

-a

(Optional) Returns all CA Workload Automation AE installation information, including the current product status.

-c

(Optional) Returns the list of all CA Workload Automation AE components installed on the computer from which you issue the command.

-d

(Optional) Returns information about the CA Workload Automation AE database.

-f

(Optional) Returns the custom (or default) CA Workload Automation AE component installation specifications.

-i

(Optional) Returns a list of all configured CA Workload Automation AE instances.

-l

(Optional) Returns the names and locations of the CA Workload Automation AE installation logs.

-p

(Optional) Returns a list of the Product Interchange File (PIF) and the RPM packages included as part of this installation.

Note: This parameter uses the RPM Package Manager and the CA lsm utility to retrieve the list. The process may take a few minutes.

-s

(Optional) Returns the current status of the CA Workload Automation AE components.

-v

(Optional) Returns the CA Workload Automation AE version, installation date and time, and installation directory.

Example: Report Default Installation Information

This example returns the CA Workload Automation AE components and version installed, the installation log locations, configured instances, and status to standard output.

```
as_info
```

Example: Report Specific Installation Information

This example returns the CA Workload Automation AE version, installation date and time, installation directory, current status, and configured instances to standard output.

```
as_info -vsi
```

as_safetool Command—Maintain Authentication Certificates

The as_safetool command maintains CA Workload Automation AE authentication certificates.

CA Workload Automation AE integrates with CA EEM, adding a comprehensive security layer. This integration results in consistent identity management, increased access security, comprehensive in-depth auditing, reduced application development costs, and improved responsiveness to business and regulatory demands. With as_safetool, you can generate authentication certificates and install or remove security policies.

Syntax

This command has the following formats:

- To run as_safetool in interactive (menu-driven) mode:

as_safetool

- To issue as_safetool from the command line:

```
as_safetool [-b CA_EEM_server]
            [-f | -h | -q | -s | -S | -x]
            [-c instance | -i instance | -u instance]
            [-a user_name | -g user_name | -r user_name ]
            [-d certificate_location]
```

-b CA_EEM_server

(Optional) Specifies the name of the CA EEM server host.

Default: localhost

Note: This value is required when you define the -c, -i, or -u options.

-f

(Optional) Forces the reinstallation of the instance policies for the CA Workload Automation AE instance defined in the -i option, if they are already installed.

Note: To use this option, you must also specify the -i option.

-h

(Optional) Displays the help for as_safetool.

-q

(Optional) Issues the as_safetool command in quiet mode. Status and error messages are not displayed.

Note: To use this option, you must also specify the -c, -i, or -u option.

-S

(Optional) Displays all instances with an installed security policy.

-S

(Optional) Displays all application instance administrators.

-X

(Optional) Returns the as_safetool version to standard output.

-c *instance*

(Optional) Specifies a CA Workload Automation AE instance. An authentication certificate is generated for that instance.

-i *instance*

(Optional) Specifies a CA Workload Automation AE instance. The instance default security policies are installed for that instance. Use this option with the -f option, which forces the reinstallation of the security policies.

-u *instance*

(Optional) Specifies a CA Workload Automation AE instance. The security policies are uninstalled for that instance.

Note: Removing the last instance deregisters CA Workload Automation AE from CA EEM.

-a *user_name*

(Optional) Defines the CA EEM server administrative account used to connect to the CA EEM back-end server.

Default: EiamAdmin

-g *user_name*

(Optional) Specifies a user name. Application instance administrative privileges are granted to that user.

-r *user_name*

(Optional) Specifies a user name. Application instance administrative privileges are removed from that user.

-d *certificate_location*

(Optional) Specifies the location to store the certificate file in.

Note: To use the -c, -g, -i, -S, or -u option, you must set the ASSAFETOOLPW environment variable to the EiamAdmin user account password.

Examples: Running as_safetool in Interactive Mode

The following examples describe how to use as_safetool in interactive mode:

Example: Deregister CA Workload Automation AE from CA EEM

Suppose that you want to deregister a CA Workload Automation AE instance with CA EEM in interactive mode. Do the following:

1. Enter the following command at the UNIX operating system prompt or the Windows command prompt:

```
as_safetool
```

The following menu appears:

```
Please select from the following options:  
[1] Manage CA EEM Backend Server Location.  
[0] Exit CA WAAE IAM Toolkit Safetool Utility.
```

2. Enter 1 and press the Enter key.

The following menu appears:

```
Please select from the following options:  
[1] Set and Connect CA EEM Backend Server Location as an Administrator.  
[2] Set CA EEM Backend Server Location.  
[3] Show current Backend Server Location.  
[9] Exit from "Manage CA EEM Backend Server Location" menu.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

3. Enter 1 and press the Enter key.

4. Enter the CA EEM back end server host name, the application instance name, and the CA EEM administrator user name and password when prompted.

5. Enter 9 and press the Enter key after connecting to the CA EEM back-end server.

The following menu appears:

```
Please select from the following options:  
[1] Manage CA EEM Backend Server Location.  
[2] Generate Instance Authentication Certificate.  
[3] Manage Instance Security Policy.  
[4] Manage Application Instance Administrators.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

6. Enter 3 and press the Enter key.

The following menu appears:

```
Please select from the following options:  
[1] Install Default Security Policy.  
[2] Uninstall Default Security Policy.  
[3] Show Instances with Security Policy Installed.  
[4] Perform Permission Checks against Instance Security Policy.  
[9] Exit from "Manage Instance Security Policy" menu.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

7. Enter 2 and press the Enter key.
8. Enter the CA Workload Automation AE instance name when prompted.

The CA Workload Automation AE instance is deregistered with CA EEM and the security policies are uninstalled.

Example: Perform Permission Checks Against a Security Policy

Suppose that you want to check a CA EEM security policy for a CA Workload Automation AE instance in interactive mode. Do the following:

1. Enter the following command at the UNIX operating system prompt or the Windows command prompt:

```
as_safetool
```

The following menu appears:

```
Please select from the following options:  
[1] Manage CA EEM Backend Server Location.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

2. Enter 1 and press the Enter key.

The following menu appears:

```
Please select from the following options:  
[1] Set and Connect CA EEM Backend Server Location as an Administrator.  
[2] Set CA EEM Backend Server Location.  
[3] Show current Backend Server Location.  
[9] Exit from "Manage CA EEM Backend Server Location" menu.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

3. Enter 1 and press the Enter key.
4. Enter the CA EEM back end server host name, the application instance name, and the CA EEM administrator user name and password when prompted.

5. Enter 9 and press the Enter key after connecting to the CA EEM back-end server.

The following menu appears:

```
Please select from the following options:  
[1] Manage CA EEM Backend Server Location.  
[2] Generate Instance Authentication Certificate.  
[3] Manage Instance Security Policy.  
[4] Manage Application Instance Administrators.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

6. Enter 3 and press the Enter key.

The following menu appears:

```
Please select from the following options:  
[1] Install Default Security Policy.  
[2] Uninstall Default Security Policy.  
[3] Show Instances with Security Policy Installed.  
[4] Perform Permission Checks against Instance Security Policy.  
[9] Exit from "Manage Instance Security Policy" menu.  
[0] Exit CA WAAE EEM Toolkit Safetool Utility.
```

7. Enter 4 and press the Enter key.

8. Enter the CA Workload Automation AE instance name when prompted.

9. Enter the security policy that you want to check when prompted. The syntax is as follows:

class resource access user

class

Specifies the policy class. Options are the following:

- as-appl
- as-calendar
- as-control
- as-cycle
- as-group
- as-gvar
- as-job
- as-joblog
- as-jobtype
- as-list
- as-machine
- as-owner
- as-resource

resource

Specifies the name of the resource policy that you want to check.

access

Specifies the type of access. Options are the following:

x

Specifies execute access.

w

Specifies write access.

r

Specifies read access.

c

Specifies create access.

d

Specifies delete access.

user

Specifies the user name.

The specified resource policy is checked.

Exit Codes for the as_safetool Command

The following exit codes are associated with the as_safetool command:

0

Indicates that the command was successful.

1

Indicates that the command contained a syntax error.

2

Indicates that the back-end server is not available or the EiamAdmin user account password is invalid.

3

Indicates that the CA Workload Automation AE registration to CA EEM failed.

4

Indicates that installation of the CA Workload Automation AE security policy failed.

5

Indicates that removal of the CA Workload Automation AE security policy failed.

6

Indicates that the unregistration of CA Workload Automation AE from CA EEM failed.

7

Indicates that the ASSAFETOOLPW environment variable is not defined.

8

Indicates that certificate generation failed.

as_server Command—Run the Application Server

Valid on UNIX

The as_server command is the application server component that runs the application server for a specific instance.

Note: The application server typically starts in the background using the unisvcntr command or boot scripts during startup.

Syntax

This command has the following format:

`as_server -A instance_name [-?] [-x]`

instance_name

Specifies the instance that is associated with the application server you want to run.

-?

(Optional) Displays help for the command.

-x

(Optional) Displays the version information.

Example: Start the Application Server

This example starts the application server for the instance ACE.

```
as_server -A ACE
```

as_test Command—Test Utility

The as_test command is a utility that can run for a specified amount of time, write a message to stdout and/or stderr, and exit with a specific exit code. When the scheduler is running in test mode to agents, Command job commands are automatically replaced with the execution of this command. You can use as_test to test job dependencies and error handling.

Syntax

This command has the following format:

```
as_test [-t Seconds] [-e Exit Code] [-c Message] [-r Message] [-x] [-?]
```

-t Seconds

(Optional) Specifies the number of seconds for the machine to sleep.

-e Exit Code

(Optional) Specifies the return code for the job to complete.

-c Message

(Optional) Specifies the message to write to the STDOUT file.

-r Message

(Optional) Specifies the message to write to the STDERR file.

-x

(Optional) Displays version information.

-?

(Optional) Displays help for the command.

Example: Run the as_test Command on a Windows machine

This example runs the as_test command on the Windows machine named winagent. The as_test command waits for 5 seconds and exits with a return code of 0.

```
insert_job: win_cmd
job_type: CMD
machine: winagent
command: as_test -t 5 -e 0
```

astail Command—Display End of File

Valid on UNIX

The astail command displays the last 10 lines of the file specified in the command.

Syntax

This command has the following format:

`astail [-f filename] [-x]`

-f *filename*

(Optional) Defines the name of the file to display.

-x

(Optional) Returns the version number of the astail command to standard output.

Note: To terminate the command, press Ctrl+C.

auto_svcdesk Command—Opens CA Service Desk Tickets

The auto_svcdesk command lets CA Service Desk helpdesk tickets to be opened on behalf of an existing job in the CA Workload Automation AE database or on behalf of some action which may occur in the CA Workload Automation AE environment. This command supplements the existing CA Service Desk integration provided by the CA Workload Automation AE scheduler component.

Syntax

This command has the following format:

```
auto_svcdesk -G | -J JobName  
              [-a Attribute/Value pair]  
              [-d Description]  
              [-i Impact]  
              [-p Priority]  
              [-s Severity]  
              [-y Summary]  
              [-L Url]  
              [-U Username -P Password | -C Customer]  
              [-?]
```

-G

Opens a generic CA Service Desk ticket request on behalf of CA Workload Automation AE.

-J *JobName*

Opens a CA Service Desk ticket request on behalf of a specific job defined to CA Workload Automation AE. The jobname specified in this option must exist in the database. The job's current attributes do not need to employ its Service Desk attributes to send a request using this command. The various options of this command can be used to set description, impact, priority, and severity. If these options are not specified, they will be set based on a pre-defined template that exists for CA Workload Automation AE in CA Service Desk.

Note: Options -G and -J are mutually exclusive.

-a *Attribute/Value pair*

(Optional) Defines CA Service Desk request attributes and handles values to be set in the CA Service Desk request. This flag is the behavioral equivalent of the svcdesk_attr job attribute.

Note: If this flag is specified for a jobname CA Service Desk request, its value overrides the job's svcdesk_attr attribute value in the database.

Use of this option requires advanced knowledge of CA Service Desk. Contact your CA Service Desk administrator for assistance.

-d Description

(Optional) Defines the message or string to be set as the description field in the CA Service Desk request. This flag is the behavioral equivalent of the svcdesk_desc job attribute.

Note: If this flag is specified for a jobname CA Service Desk request, its value overrides the job's svcdesk_desc attribute value in the database.

-i Impact

(Optional) Specifies the impact level to assign to the CA Service Desk request. This flag is the behavioral equivalent of the svcdesk_imp job attribute.

Note: If this flag is specified for a jobname CA Service Desk request, its value overrides the job's svcdesk_imp attribute value in the database.

-p Priority

(Optional) Specifies the priority level to assign to the CA Service Desk request. This flag is the behavioral equivalent of the svcdesk_pri job attribute.

Note: If this flag is specified for a jobname CA Service Desk request, its value overrides the job's svcdesk_pri attribute value in the database.

-s Severity

(Optional) Specifies the severity level to assign to the CA Service Desk request. This flag is the behavioral equivalent of the svcdesk_sev job attribute.

Note: If this flag is specified for a jobname CA Service Desk request, its value overrides the job's svcdesk_sev attribute value in the database.

-y Summary

(Optional) Defines the message or string to be set as the summary field in the CA Service Desk request. There is no job CA Service Desk attribute behavioral equivalent. The scheduler component when interfacing with CA Service Desk has a pre-defined summary, which it passes as part of the request.

-L Url

(Optional) Specifies the URL of the CA Service Desk web server. If this option is not specified, the URL is obtained from the CA Workload Automation AE environment (derived from the current CA Workload Automation AE integration settings).

Note: For more information about CA Workload Automation AE integration settings, see the *User Guide*.

-U Username

(Optional) Specifies a valid login identifier to access the CA Service Desk web server. If this option is not specified, the login identifier is obtained from the CA Workload Automation AE environment (derived from the current CA Workload Automation AE integration settings).

Note: For more information about CA Workload Automation AE integration settings, see the *User Guide*.

-P Password

(Optional) Specifies the password for the CA Service Desk login identifier. You must specify this option, if the -U option has been specified.

-C Customer

(Optional) Specifies a valid customer identifier to access the CA Service Desk web server. If this option is not specified, the customer identifier is obtained from the CA Workload Automation AE environment (derived from the current CA Workload Automation AE integration settings).

Note: For more information about CA Workload Automation AE integration settings, see the *User Guide*.

-?

(Optional) Displays the help for auto_svcdesk command.

Example: Open a CA Service Desk Ticket

This example opens a CA Service Desk ticket for a job named ‘svcdesk_test’.

```
auto_svcdesk -J svcdesk_test
```

Example: Open a CA Service Desk Ticket and Set the Priority and Impact Levels

This example opens a CA Service Desk ticket for a job named ‘svcdesk_test’ and sets the priority level to a value of 1 and the impact level to a value of 4:

```
auto_svcdesk -J svcdesk_test -p 1 -i 4
```

Note: This command overrides the jobs current priority and impact, if set in the database.

Example: Open a CA Service Desk Ticket and Set the Description

This example opens a CA Service Desk ticket for a job named ‘svcdesk_test’ and sets the description to ‘This is a test’ and the urgency to a value of 5:

```
auto_svcdesk -J svcdesk_test -d "This is a test" -a "urgency/urg:1104"
```

Note: This command overrides the jobs current description and impact, if set in the database.

Example: Open a CA Service Desk Ticket and Direct it to a Specific Web Server

This example opens a CA Service Desk ticket for a job named ‘svcdesk_test’ and directs it to a specific CA Service Web Server named ‘unisvcdesk’:

```
auto_svcdesk -J svcdesk_test -L  
http://unisvcdesk:8080/axis/services/USD_R11_WebService
```

Example: Open a CA Service Desk Ticket to Identify a Problem

This example opens a generic CA Service Desk ticket to identify a problem with a database rollover, where the priority is 1:

```
auto_svcdesk -G -y "CA Workload Automation AE DB_ROLLOVER" -d "Database rollover has occurred. Product currently running off secondary database. Primary needs to be investigated." -p 1
```

This request can be initiated from an ALARM message forwarded to CA NSM Event Management, where the message action to be executed is the auto_svcdesk command.

More information:

[JIL Job Definitions](#) (see page 287)

autoaggr Command—Aggregate Data

The autoaggr command aggregates CA Workload Automation AE data into hourly, daily, weekly, and monthly tables that other programs use to generate reports. The CA Workload Automation AE SDKs provide APIs to retrieve the data collected by the autoaggr command.

In all cases, the product aggregates data in smaller intervals before starting the specified aggregation. For example, when you specify -d (daily), the product aggregates the hourly data before starting the daily aggregation.

Running autoaggr during non-critical times significantly reduces the time required to generate reports because the necessary data points are readily available without on-demand generation.

Note: The autoaggr command can run on any computer where a database client is configured to communicate to the CA Workload Automation AE event servers and where CA Workload Automation AE is properly configured.

Syntax

This command has the following format:

```
autoaggr [-d] [-h] [-m] [-w] [-r] [-x] [-?]
```

-d

(Optional) Aggregates daily data.

-h

(Optional) Aggregates hourly data.

-m

(Optional) Aggregates monthly data.

-w

(Optional) Aggregates weekly data.

-r

(Optional) Deletes the current aggregated data, and reaggregates data.

-x

(Optional) Returns the autoaggr version number to standard output.

-?

(Optional) Displays help for the command.

Example: Aggregate Hourly Data

This example aggregates hourly data.

```
autoaggr -h
```

Example: Aggregate Hourly and Daily Data

This example aggregates hourly and daily data.

```
autoaggr -d
```

Example: Aggregate Hourly, Daily, Weekly, and Monthly Data

This example aggregates hourly, daily, weekly, and monthly data.

```
autoaggr -m
```

autocal_asc Command—Manage Calendars

The autocal_asc command is a client component utility that provides a text-based command line mechanism for adding, deleting, printing, exporting, and importing calendars. You can use autocal_asc to import calendars exported with the autocal command in previous CA Workload Automation AE releases.

CA Workload Automation AE uses calendars to schedule the days on which jobs should run. Each calendar has a unique name and contains a list of days on which the jobs may run or are excluded from running.

The autocal_asc command can manipulate the following types of calendars:

- Standard calendars
- Extended calendars
- Cycles

Note: Calendar names can be up to 30 characters and can **only** include the following characters: a-z, A-Z, 0-9, period (.), underscore (_), pound (#), and hyphen (-).

When you invoke autocal_asc with CA EEM security enabled, CA Workload Automation AE uses the following rules to verify user permissions based on the activity selected:

- Read access is required to print the autocal_asc version number or to export a calendar.
- Create access is required to add or import a new calendar or cycle.
- Write access is required to modify an existing calendar. Adding periods to an existing cycle requires both read and write access.
- Delete access is required to delete a calendar or cycle.

After creating a calendar, you can use one of the following methods to reference it in your job definitions:

- With jil, enter a calendar name in the run_calendar or exclude_calendar attribute.
- In the CA WCC GUI, enter a calendar name in the appropriate field.

When you update a calendar, CA Workload Automation AE recalculates the starting times for all jobs that use the updated calendar.

Note: You must regenerate an extended calendar to reflect the changes made.

Syntax

This command has the following formats:

- To run autocal_asc in interactive (menu-driven) mode:
`autocal_asc`
- To delete a calendar or cycle:
`autocal_asc -D {-c cycle_name | -e extcal | -s stdcal}`
- To export a calendar or cycle:
`autocal_asc -E export_filename [-c cycle_name | -e extcal | -s stdcal]`
- To import calendars or cycles:
`autocal_asc -I import_filename`
- To list calendar or cycle names:
`autocal_asc -l | -lc | -le | -ls`
- To display the version number:
`autocal_asc -x`
- To display help for the command:
`autocal_asc -?`

-D

(Optional) Deletes the calendar or cycle specified by **-e extcal**, **-c cycle_name**, or **-s stdcal** from CA Workload Automation AE.

- To delete an extended calendar, specify **-D -e extcal**.
- To delete a cycle, specify **-D -c cycle**.
- To delete a standard calendar, specify **-D -s stdcal**.

-E export_filename

(Optional) Exports the calendar or cycle specified by **-e extcal**, **-c cycle_name**, or **-s stdcal** to the file defined in the *export_filename* parameter.

- To export an extended calendar definition, specify **-E export_filename -e extcal**. This exports the extended calendar definition.
- To export the list of dates generated by an extended calendar, specify **-E export_filename -s stdcal**. This output file resembles a standard output.
- To export a cycle, specify **-E export_filename -c cycle_name**.
- To export a standard calendar, specify **-E export_filename -s stdcal**.
- To export all calendars of a specific type, specify **-E export_filename** followed by **-s ALL**, **-e ALL**, or **-c ALL**.

-e extcal

(Optional) Defines an external calendar to export or delete.

- To export an external calendar, specify **-e extcal -E export_filename**.
- To delete an external calendar, specify **-e extcal -D**.

-c cycle_name

(Optional) Defines a cycle to export or delete.

- To export a cycle, specify **-c cycle_name -E export_filename**.
- To delete a cycle, specify **-c cycle_name -D**.

-s stdcal

(Optional) Specifies a standard calendar to export or delete.

- To export a standard calendar, specify **-s stdcal -E export_filename**.

To delete a standard calendar, specify **-s stdcal -D**.

-I import_filename

(Optional) Imports calendars or cycles from the specified file.

-l

(Optional) Lists all extended and standard calendars.

-ls

(Optional) Lists all standard calendars.

-le

(Optional) Lists all extended calendars.

-lc

(Optional) Lists all cycles.

-

(Optional) Reads menu options from standard input, processes options, and terminates.

Example: Import Calendars from a File

This example imports the calendars defined in the file federal_holidays.txt.

```
autocal_asc -I federal_holidays.txt
```

Example: Export a Standard Calendar to a File

This example exports the standard calendar federal_holidays to the file holidays_06.txt.

```
autocal_asc -s federal_holidays -E holidays_06.txt
```

Example: Export a Cycle to a File

This example exports the cycle REPORT_PERIOD to the file 06_cycles.txt.

```
autocal_asc -c REPORT_PERIOD -E 06_cycles.txt
```

Example: Delete a Standard Calendar

This example deletes the standard calendar Holiday2011.

```
autocal_asc -D -s Holiday2011
```

Example: List all Calendar Names

This example lists all standard and extended calendar names.

```
autocal_asc -l
```

Example: View autocal_asc Version Information

This example returns the version information for autocal_asc.

```
autocal_asc -x
```

Example: Define a Standard Calendar

The following is a calendar definition for the standard calendar, Holiday2006. This calendar can be used with jil attributes run_calendar or exclude_calendar as well as in an extended calendar definition.

```
calendar: Holiday2006
01/01/2006 00:00
01/02/2006 00:00
01/16/2006 00:00
02/20/2006 00:00
05/29/2006 00:00
07/02/2006 00:00
09/04/2006 00:00
11/23/2006 00:00
11/24/2006 00:00
12/25/2006 00:00
12/26/2006 00:00
```

This example schedules jobs to run everyday of the week at 2:00 p.m., except for the days listed in the Holiday2006 calendar.

```
insert_job:JobWithExcludeCalendar
command:ls
machine:localhost
date_conditions:1
days_of_week:all
start_times:"14:00"
exclude_calendar:Holiday2006
```

Example: Define an Extended Calendar

The following extended calendar definition creates a calendar using the Quarter2007 cycle calendar listed below. It sets the workdays as Monday through Friday, which is the default. The non workday action field “W” indicates that jobs scheduled to run on a non workday will instead run on the next work day. The condition CWEEK#L schedules jobs to run on the last week of each period in the cycle. In this example, jobs are scheduled to run in the last week of March, June, September, and December. Jobs that fall on a Saturday or Sunday are scheduled to run on the following Monday.

```
extended_calendar: LastWeekCycleCal
workday: mo,tu,we,th,fr
non_workday: W
holiday:
holcal:
cyccal: Quarter2007
adjust: 0
condition: CWEEK#L
```

This example creates an extended calendar for the day before the first Wednesday of the month. The WED#1 keyword indicates the first Wednesday of the month and the -1 adjustment field sets it to the day before that Wednesday.

```
extended_calendar: FIRST-WED-MINUS-1
workday: mo,tu,we,th,fr
non_workday:
holiday:
holcal:
cyccal:
adjust: -1
condition: WED#1
```

Example: Define a Cycle Calendar

This example creates the cycle calendar, Quarter2007, with 4 periods.

```
cycle: Quarter2007
start_date: 01/01/2007
end_date: 03/31/2007
start_date: 04/01/2007
end_date: 06/30/2007
start_date: 07/01/2007
end_date: 09/30/2007
start_date: 10/01/2007
end_date: 12/31/2007
```

Standard Calendars

A standard calendar lists user-specified dates. Each entry in a standard calendar consists of a date and (optionally) a time.

Extended Calendars

An extended calendar uses rules to generate a list of dates equivalent to a standard calendar. You can use either a standard calendar or an extended calendar as the run_calendar or exclude_calendar attributes of a job definition.

When you create an extended calendar, it includes each date in the coming year that obeys the rules you specify. When those dates are exhausted, the extended calendar automatically applies the rules again if necessary to determine the next run time of a job.

When an extended calendar generates its list of dates, it considers 368 days beginning on the day before the current date. This time factor ensures a full year in all time zones, including leap years. For each day in the year, the extended calendar checks the rules in three steps. First, it evaluates the keywords in the condition field. If the result is true, the day is a candidate for inclusion in the calendar. Next, the extended calendar applies the actions in the holiday and non_workday parameters. Finally, it adds the number of days in the adjust parameter and includes the resulting date in the list.

To avoid errors, preview the dates generated from the rules of your extended calendar. Typographical errors can result in no dates or the wrong ones.

You can use the following parameters to define rules:

workday

Specifies which days of the week are workdays with respect to the non_workday parameter and the WORK keywords in the condition field below. On the command line, workdays can be specified as: "mo", "tu", "we", "th", "fr", "sa", "su". They can also be specified by [XXXXX..], where Monday corresponds to the first "X". An "X" indicates that the corresponding day is a workday; "." states that the corresponding day is not a workday.

non_workday

Indicates the action to take when the selected day falls on a non-workday. Valid values are as follows:

- blank—Include the day whether or not it is a workday.
- O—Include only non-workdays in the calendar.
- N—Include the next day, which could be a non-workday.
- W—Include the next workday.
- P—Include the previous workday.

holiday

Indicates the action to take if the selected day is in the calendar named in the holcal parameter. Valid values are as follows:

- blank—Do not include holidays.
- O—Include only holidays.
- S—Include the day whether or not it is a holiday.
- N—Include the next day, which could be a holiday.
- W—Include the next workday.
- P—Include the previous workday.

holcal

Specifies the name of a standard calendar to use for holiday dates in the holiday parameter and whether the condition parameter contains the HOLIDAYS keyword.

cycical

Defines the cycle name to use and is only valid when the cycle keywords appear in the Extended Calendar conditions.

adjust

Defines the number of days before (negative number) or after (positive number) the other specified criteria to run an associated job. Use adjustments if the other rules do not define the necessary days. For example, to run a job the day after every holiday, you could enter "holidays" in the Extended Calendar conditions field and "+1" in the Adjustments field.

condition

Defines the principal rule for choosing dates to include in the extended calendar.

You can enter compound conditions like "mon or tue" and complex expressions like "holidays and (mon or tue)". The following operators are available, in decreasing order of precedence:

- **NOT** or **!** or **^**—Include a day if the following expression would not include it.
- **AND** or **&**—Include a day if the expressions on both sides of AND include it.
- **OR** or **|**—Include a day if an expression on either side of OR includes it.

You can use parentheses **()** to override the normal precedence.

Be careful not to confuse these logical operators with English. For example, write "mon or tue" if you want a job to run on Mondays and Tuesdays. The expression "mon and tue" includes no dates, since no day is both a Monday and a Tuesday.

In the following keywords, uppercase letters and the # symbol are constants, and the lowercase letters are replaced with a day or week value. You can enter the keywords in either upper or lower case.

Keywords ending in M1 (first from the end) have the same meaning as the ones ending in #L (last). If you write conditions in lower case, use #01 and m1 to avoid confusing #1 (first) with #l (last).

Abbreviations for months and days of the week in the condition keywords use English spelling in all locales. Explicit dates in the condition field are written year/month/day irrespective of the DateFormat configuration variable or the locale.

You can use the following condition keywords:

- **DAILY**—Include every day. This keyword is the default if the condition field is empty.
- **HOLIDAYS**—Include every day that is in the calendar named in the holcal parameter.
- **ddd**—Include the specified day of the week. Replace *ddd* with a three-character abbreviation for the day (mon, tue, wed, thu, fri, sat, or sun).
- **ddd#n**—Include the *n*th occurrence of day *ddd* in the month. Replace *ddd* with a three-character abbreviation for the day (mon, tue, wed, thu, fri, sat, or sun). Replace *n* with a number in the range 1 to 5 or L for the last occurrence in the month.

- **Xddd#n**—Include all occurrences of day *ddd* (mon - sun) in a month except the *n*th one. Replace *n* with a number in the range 1 to 5 or L for the last occurrence in the month.
- **dddMn**—Include the *n*th occurrence of day *ddd* (mon - sun) from the end of the month. Replace *n* with a number in the range 1 to 5.
- **XdddMn**—Include all occurrences of day *ddd* (mon - sun) in a month except the *n*th one from the end of the month. Replace *n* with a number in the range 1 to 5.
- **WORKDAYS**—Include every day defined in the workday parameter but not in the holiday calendar specified in the holcal parameter. Dates in the holiday calendar are never considered workdays, irrespective of the action you specify in the holiday parameter.
- **WORKD#nn**—Include the *nn*th workday of the month. Replace *nn* with a number in the range 1 to 31 or L for the last day of the month.
- **WORKDMnn**—Include the *nn*th workday from the end of the month. Replace *nn* with a number in the range 1 to 31.
- **WORKDXnn**—Include all workdays of the month except the *nn*th one. Replace *nn* with a number in the range 1 to 31 or L for the last day of the month.
- **FOMWORK**—Include the first workday of the month.
- **EOMWORK**—Include the last workday of the month.
- **XFOMWORK**—Include all workdays of the month except the first one.
- **XEOMWORK**—Include all workdays of the month except the last one.
- **WEEKDAYS**—Include Monday through Friday except for holidays. Dates in the holiday calendar are never considered weekdays, irrespective of the action you specify in the holiday parameter.
- **WEEKD#n**—Include the *n*th weekday. Replace *n* with a number in the range 1 to 5 or L for the last weekday.
- **WEEKDMn**—Include the *n*th weekday from the end of the week. Replace *n* with a number in the range 1 to 5.
- **WEEKDXn**—Include all weekdays except the *n*th one. Replace *n* with a number in the range 1 to 5 or L for the last weekday.
- **FOMWEEK**—Include the first weekday of the month.
- **EOMWEEK**—Include the last weekday of the month.
- **XFOMWEEK**—Include all weekdays of the month except the first one.
- **XEOMWEEK**—Include all weekdays of the month except the last one.

- **WEEK#nn**—Include the nnth week of the year. Each week begins on the same day of the week as January 1st. The last week has one day, or two days in a leap year. Replace nn with a number in the range 1 to 53 or L for the last week of the year.
- **WEEKXnn**—Include all weeks of the year except the nnth one. Each week begins on the same day of the week as January 1st. The last week has one day, or two days in a leap year. Replace nn with a number in the range 1 to 53 or L for the last week of the year.
- **WEEKMnn**—Include the nnth week from the end of the year. Each week begins on the same day of the week as January 1st. The last week has one day, or two days in a leap year. Replace nn with a number in the range 1 to 53.
- **WEKRd#nn**—Include the nnth full week of the year, starting on a day that you specify. The last week of the year may begin in December and end in January. Replace the optional d with the first day of the week (for example, 1 for Monday or 7 for Sunday). If you omit d, Monday is the first weekday. Replace nn with a number in the range 1 to 52 or L for the last week.
- **WEKRdXnn**—Include all full weeks of the year, starting on a day you specify, except the nnth week. The last week of the year may begin in December and end in January. Replace the optional d with the first day of the week (for example, 1 for Monday or 7 for Sunday). If you omit d, Monday is the first weekday. Replace nn with a number in the range 1 to 52 or L for the last week.
- **WEKRdMnn**—Include the nth full week from the end of the year, starting on a day that you specify. The last week of the year may begin in December and end in January. Replace the optional d with the first day of the week (for example, 1 for Monday or 7 for Sunday). If you omit d, Monday is the first weekday. Replace nn with a number in the range 1 to 52 or L for the last week.
- **FOM**—Include the first day of the month.
- **EOM**—Include the last day of the month.
- **XFOM**—Include all days of the month except the first one.
- **XEOM**—Include all days of the month except the last one.
- **MNTHD#nn**—Include the nnth day of the month. Replace nn with a number in the range 1 to 31 or L for the last day of the month.
- **MNTHDMnn**—Include the nnth day from the end of the month. Replace nn with a number in the range 1 to 31.
- **MNTHDXnn**—Include all days of the month except the nnth one. Replace nn with a number in the range 1 to 31 or L for the last day of the month.
- **mmm**—Include the specified month. Replace mmm with a three-character abbreviation of the month name (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec).

- **mmm#nn**—Include the *nn*th day of month *mmm*. Replace *mmm* with a three-character abbreviation of the month name (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec). Replace *nn* with a number in the range 1 to 31 or L for the last day of the month.
 - **Xmmm#nn**—Include all days of month *mmm* except the *nn*th one. Replace *mmm* with a three-character abbreviation of the month name (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec). Replace *nn* with a number in the range 1 to 31 or L for the last day of the month.
 - **mmmMnn**—Include the *nn*th day from the end of month *mmm*. Replace *mmm* with a three-character abbreviation of the month name (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec). Replace *nn* with a number in the range 1 to 31.
 - **XmmmMnn**—Include all days of month *mmm* except the *nn*th day from the end. Replace *mmm* with a three-character abbreviation of the month name (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec). Replace *nn* with a number in the range 1 to 31.
 - **yy/mm/dd**—Include the specified year, month, and day.

Replace *yy* with a number in the range 00 to 99, **n* (where the year can be any year ending in *n*), *n** (where the year can be any year beginning in *n*), or ** (any year). Years less than 80 are assumed to occur in the twenty-first century.

Replace *mm* with a number in the range 01 to 12, **n* (where the month can be any month ending in *n*), *n** (where the month can be any month beginning in *n*), or ** (any month).

Replace *dd* with a number in the range 01 to 31, **n* (where the day can be any day ending in *n*), *n** (where the day can be any day beginning in *n*), or ** (any day).
- For example:
- **/01/01—Include January 1st of every year.
- 96/**/02—Include the 2nd day of every month in 1996.
- 96/03/*3—Include March 3, 13, and 23 in 1996.
- **yyyy/mm/dd**—Include the specified year, month, and day.

Replace *yyyy* with a number in the range 1900 to 2100, **n* (where the year can be any year ending in *n*), *n** (where the year can be any year beginning in *n*), or ** (any year).

Replace *mm* with a number in the range 01 to 12, **n* (where the month can be any month ending in *n*), *n** (where the month can be any month beginning in *n*), or ** (any month).

Replace *dd* with a number in the range 01 to 31, **n* (where the day can be any day ending in *n*), *n** (where the day can be any day beginning in *n*), or ** (any day).

You can use the following keywords in the condition parameter when you specify a cycle name in the cycal parameter:

- **CYCLE**—Include any day within the date ranges, or periods defined in the cycle.
- **CYCL#nnn**—Include a specific day of each period in the cycle. Replace *nnn* with a number in the range 1 to 365.
- **CYCL#L**—Include the last day of each period in the cycle.
- **CYCLMnnn**—Include a specific day from the end of each period in the cycle. Replace *nnn* with a number in the range 1 to 365.
- **CYCLXnnn**—Include all days except day *nnn* of each period in the cycle. Replace *nnn* with a number in the range 1 to 365.
- **CYCP#nn**—Include a specific period in the cycle. Replace *nn* with a number in the range 1-36.
- **CWORKD**—Include all workdays in the cycle date range.
- **CWEEK#nn**—Include a specific week of each period in the cycle. The week begins on the first day of the period. The last week of a period may be a partial week. Replace *nn* with a number in the range 1 to 52.
- **CWEEK#L**—Include the last week of each period in the cycle. The last week of a period may be a partial week.
- **CWEEKMnn**—Include a specific week from the end of each period in the cycle. Replace *nn* with a number in the range 1 to 52.
- **CWEEKXnn**—Include all weeks except week *nn* of each period in the cycle. Replace *nn* with a number in the range 1 to 52.
- **CWEEKD**—Include all weekdays in the cycle date range.
- **CWRK#nnn**—Include a specific workday of each period in the cycle. Replace *nnn* with a number in the range 1 to 300.
- **CWRK#L**—Include the last workday of each period in the cycle.
- **CWRKMnnn**—Include a specific workday from the end of each period in the cycle. Replace *nnn* with a number in the range 1 to 300.
- **CWRKXnnn**—Include all workdays except day *nnn* of each period in the cycle. Replace *nnn* with a number in the range 1 to 300.
- **CWRKXL**—Include every workday except the last workday of each period in the cycle.
- **CWEK#n**—Include a specific weekday of each period in the cycle. Replace *n* with a number in the range 1 to 5.
- **CWEK#L**—Include the last weekday of each period in the cycle.
- **CWEKMn**—Include a specific weekday from the end of the week of each period in the cycle. Replace *n* with a number in the range 1 to 5.

- **CWEKXn**—Include all weekdays except day n of each period in the cycle. Replace *n* with a number in the range 1 to 5.
- **Cddd#nn**—Include a specific occurrence of day *ddd* of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN). Replace *nn* with a number in the range 1 to 52.
- **Cddd#L**—Include the last occurrence of the day of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN).
- **CdddMnn**—Include the specific occurrence of the day from the end of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN). Replace *nn* with a number in the range 1 to 52.
- **XCddd#nn**—Include all occurrences of the specified day except the *nn*th one of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN). Replace *nn* with a number in the range 1 to 52.
- **XCddd#L**—Include all occurrences of the specified day except the last one of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN).
- **XCdddMnn**—Include all occurrences of the specified day except the *nn*th one from the end of each period in the cycle. Replace *ddd* with the three-character abbreviation of the day (MON, TUE, WED, THU, FRI, SAT, or SUN). Replace *nn* with a number in the range 1 to 52.

For example:

CYCL#44—Include day 44 of each period in the cycle.

CWEEKM05—Include the fifth week from the end of each cycle period.

CWRK#L—Include the last workday of each period in the cycle.

XCSUN#52—Include all Sundays specified in the cycle except for Sunday of week 52 of each period.

Cycles

A cycle is a list of one or more date ranges used when cycle keywords are present in an extended calendar definition. Each range or period consists of a start date and end date. The start date is the first day of the period. The end date is the last day of the period (this value must be greater than or equal to the start date of the period). Valid values are dates that follow the format specified in CA Workload Automation AE.

The year 1972 has a special wildcard meaning. Cycle periods that start and end in the year 1972 repeat automatically on the same dates every year. In this case, the end date may be before the start date, meaning that the period begins in one year and ends in the next.

Note: A cycle can have up to 30 periods. Each period can be at most 366 days in length. Cycle names can be up to 30 characters long and can **only** include the following characters: a-z, A-Z, 0-9, period (.), underscore (_), pound (#), and hyphen (-).

autoflags Command—Return System Information

The autoflags command is an agent, client, scheduler, and server component utility that returns information about the CA Workload Automation AE release number, the database being used, and the operating system. You can also use autoflags to verify the proper host name and host ID for license key generation.

Syntax

This command has the following format:

```
autoflags -a | -i | -o | -d | -v | -x | -r | -h | -n | -?  
-a  
      (Optional) Writes all autoflags information to standard output.  
-i  
      (Optional) Writes the tape ID number to standard output.  
-o  
      (Optional) Writes the operating system information to standard output.  
-d  
      (Optional) Writes the database type (SYB for Sybase, MSQ for MSSQL, or ORA for Oracle) to standard output.
```

-v

(Optional) Writes the version number to standard output.

-x

(Optional) Writes the product release number to standard output.

-r

(Optional) Writes the product version, release, and tape ID number to standard output.

-h

(Optional) Writes the host ID to standard output.

Note: The Windows operating system does not employ host IDs; therefore, on a Windows computer, the value displayed is 0.

-n

(Optional) Writes the host name to standard output.

-?

(Optional) Displays help for the command.

Note: You can specify multiple options at one time.

Example: Return System Information

This example returns all CA Workload Automation AE and system configuration information to standard output.

`autoflags -a`

On UNIX, the command displays information similar to the following in standard output:

`3 AIX ANY 11.3 0 c0a9e38d venice`

On Windows, the command displays information similar to the following in standard output:

`5 Windows_NT ANY 11.3 0 0 venice`

autoping Command—Verify Server, Agent, and Client Communication

The autoping command is a client component utility that verifies that the server and agent computers are properly configured and are communicating successfully. It also verifies that the agent is able to communicate with the application server. If you are running multiple application servers, autoping verifies that the agent is able to communicate with all the application servers. When requested, the command generates an alarm if it detects problems.

Because the client/server communication facilities are critical, autoping provides valuable information for troubleshooting. We recommend that you use autoping early in the troubleshooting process.

You can issue autoping from any machine where the autoping executable resides, and the target can be any machine that has a valid machine definition defined.

Syntax

This command has the following format:

```
autoping {-m machine | -m ALL} [-A] [-S] [-x] [-?]
```

-m *machine* | -m ALL

Specifies a valid machine to verify. The machine must be defined to the database and accessible over the network. Specify **-m ALL** to verify all machines.

Note: If you specify **-m localhost**, the scheduler must have associated a valid machine definition to localhost on startup. If localhost is not associated with a valid machine, the autoping command fails.

-A

(Optional) Sends an alarm when errors occur.

-S

Tests the connectivity between the application server and the specified agent.

Note: If you issue **autoping -M *machine* -S** against the legacy agent (machine type r, n, L, or I), the command reverts to its r11 behavior and tests the database connectivity between the event server and the legacy agent.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Notes:

- When you issue the autoping command to a machine of type 'a', the client (the machine from which you issued autoping) sends a request to the application server and waits for the application server to respond. The application server contacts the scheduler and notifies it to ping the agent and waits for the scheduler to respond. The application server then pings the agent and prepares a response to autoping. If successful, the autoping command writes the following message to standard output on the server:

```
AutoPinging Machine [machine]  
AutoPing WAS SUCCESSFUL!
```

- When you issue the autoping command to a machine of type 'r', 'n', 'l', or 'L' (legacy agent), the client (the machine from which you issued autoping) establishes a connection with the legacy agent and waits for the legacy agent to respond. If successful, the autoping command writes the following message to standard output on the server:

```
AutoPinging Machine [machine]  
AutoPing WAS SUCCESSFUL!
```

- If there is a configuration problem, the autoping command writes a message indicating that the remote machine did not respond or that a more serious problem (such as a socket read error) exists. It also writes messages on behalf of the scheduler and the application server.

Example: Verify Machine Configuration

This example verifies that the machine named myhost is properly configured and that its agent can function properly.

```
autoping -m myhost
```

If successful, the following displays:

```
CAUAJM_I_50023 AutoPinging Machine [myhost]  
CAUAJM_I_50025 AutoPing WAS SUCCESSFUL.
```

Example: Verify Agent and Application Server Connectivity

This example verifies agent and application server connectivity for all CA Workload Automation Agents defined in the event server.

```
autoping -m ALL -S
```

If successful, the following is displayed:

```
CAUAJM_I_50023 AutoPinging Machine [myhost1]
CAUAJM_I_50031 Checking the Agent's connectivity to the Application Server.
CAUAJM_I_50025 AutoPing WAS SUCCESSFUL.
```

```
CAUAJM_I_50023 AutoPinging Machine [myhost2]
CAUAJM_I_50031 Checking the Agent's connectivity to the Application Server.
CAUAJM_I_50025 AutoPing WAS SUCCESSFUL.
```

autoprofm Command—Convert Job Profiles to the New Format (Windows Only)

Valid on Windows

In Unicenter AutoSys JM r4.5 and r11, job profile information was stored in the Windows registry. To upgrade to r11.3, the job profile information in the registry must be converted to a file format that is compatible with the new CA Workload Automation Agent for UNIX, Linux, or Windows.

When you upgrade CA Workload Automation AE, the upgrade process automatically converts the job profiles. You can also issue the autoprofm command to manually convert profiles after an upgrade. You can specify where these converted files are stored.

Note: On UNIX, the job profiles are still defined in shell script files that contain the environment variables you want to source. You can specify the name of the script file (or the path to and name of the file) in the profile attribute.

Syntax

This command has the following format:

```
autoprofm -P directory [-N agent_name] [-x] [-?]
```

-P *directory*

Specifies the directory to store the converted profiles.

-N *agent_name*

(Optional) Specifies the name of an agent.

Default: WA_AGENT

-x

(Optional) Displays the version information.

-?

(Optional) Displays the command help.

Notes:

- If you are converting profiles from different r4.5 instances and the profiles have the same name, the autoprofm command renames the files to *profilename_instancename* to avoid overwriting the files.
- If you are converting profiles from different r11 instances and the profiles have the same name, the autoprofm command renames the files to *profilename_11.0_Base* to avoid overwriting the files.
- Similarly, if the r11.3 instance already has job profiles with the same name, the autoprofm tool renames the migrated files to *profilename_instancename* or *profilename_11.0_Base*. The files are renamed to avoid overwriting the existing job profiles.
- To convert the DEFAULT job profile, the autoprofm command collects the environment variables in all the DEFAULT profiles and writes them a single DEFAULT profile for CA Workload Automation AE r11.3. The agent sets the oscomponent.environment.variables parameter in its agentparm.txt file to this new DEFAULT profile. Jobs that do not have a profile associated with them uses the environment variables defined in the new DEFAULT profile.
- The migration activity is logged in the %AUTOUSER%\out\autoprofm.*instance_name* file.

autorep Command—Report Job, Machine, and Variable Information

The autorep command is a client component utility that generates reports about objects such as jobs, machines, and global variables currently defined in the database. The autorep command retrieves data from the database to formulate the reports. You can use autorep to do the following:

- Display a summary of all currently defined jobs
- Display current machine load information
- List all relevant event information for the last run of any job or for a specified job run
- Back up job definitions by extracting definitions and saving them to an output file
- List objects (for example, blobs, external instances, job, user-defined job types, global variables, machines, and resources) and their definitions.

Note: You can only view reports for the objects that you have read access to.

Syntax

This command has the following formats:

- To report on a job:

```
autorep -J job_name [-d | -s | -q | -o overnum]
[-n] [-w] [-t] [-R run_num] [-L print_level]
[-a] [-f blobfilepath]
```
- To report on a user-defined job type:

```
autorep -Y job_type [-d | -s | -q] [-n] [-w]
```
- To report on a machine:

```
autorep -M machine [-p | -q] [-d | -s] [-n] [-w]
```
- To report on jobs in a group:

```
autorep -B group [-d | -s | -q | -o overnum]
[-I application] [-n] [-w] [-t]
[-R run_num] [-L print_level]
```
- To report on jobs in an application:

```
autorep -I application [-d | -s | -q | -o overnum]
[-B group] [-n] [-w] [-t]
[-R run_num] [-L print_level]
```
- To report on global variables:

```
autorep -G global_name [-n] [-w] [-d | -s]
```

- To report on an external instance:

```
autorep -X ext_instance [-s | -q] [-n] [-w] [-t]
```

- To report on a virtual resource:

```
autorep -V virtual_resource_name [-d | -s | -q] [-n] [-w]
```

- To report on a global blob:

```
autorep -z globalblob_name [-d | -s | -q] [-n] [-w]  
[-a] [-f blobfilepath]
```

- To display the version number for the command:

```
autorep -x
```

- To display help for the command:

```
autorep -?
```

-J *job_name*

(Optional) Defines the job to report on. This value allows wildcard characters. To report on all jobs, specify -J ALL.

-M *machine*

(Optional) Specifies the machine to report on. To generate a report about all machines, specify -M ALL.

-G *global_name*

(Optional) Generates a global variable report, where *global_name* is the name of a global variable set with the sendevent command. This value allows wildcard characters. To generate a report about all global variables, specify -G ALL.

-s

(Optional) Generates a summary report. If you do not specify a report type to generate, CA Workload Automation AE generates a summary report by default.

This report contains the following information:

- For external instances, the summary report contains the name, type, machines, and port.
- For globs, the summary report contains the glob name and file associated with the glob.
- For jobs, the summary report contains the job name, start date and time, end date and time, current status, run number, and priority.

When you specify **-r runnum**, the report contains information for the specified run; when you omit **-r runnum**, the report contains information for the most recent run.

- For machines, the summary report contains the machine name, maximum load, current load, factor, type, and status.
- For user-defined job types, the summary report contains the job type, command, and description.
- For virtual resources, the summary report contains the resource name, type, machine, and defined and available units.

Note: This parameter is ignored for global variable reports.

-n

(Optional) Reports truncated names.

By default, the autorep command displays the full, non-truncated names of objects such as jobs, machines, resources, variables, and blobs. When you specify **-n**, the report displays narrow columns, which may truncate names.

-w

(Optional) Reports the long form of job conditions.

By default, the autorep command displays the short form of conditions, for example, **condition: s(JobA)**. When you specify **-w**, the report displays the long form of conditions, for example, **condition: SUCCESS(JobA)**.

-d

(Optional) Generates a detail report.

This report contains the following information:

- For jobs, the report covers all events from the most recent run of the specified jobs (the default). The report contains the following data for each event: status, the date and time of the event, try number, event state, process date and time, machine, whether the job ran with an override, and the override number.
- For machines, the report contains the following data for each machine: machine name, maximum load, current load, factor, type, and status. For jobs currently running on each machine, the report also contains the following data: job name, machine, status, load, and priority.
- For user-defined job types, the report contains the job type and the jobs that reference them.
- For virtual resources, the report contains the resource name, type, machine, defined and available units, and the jobs currently using the resource.

Note: This parameter is ignored for global variable reports, glob reports, and external instance reports.

-q

(Optional) Generates a query report, which contains the current JIL definition for the object.

Note: This parameter is ignored for global variable reports.

-o overnum

(Optional) Generates an override report, which contains the overrides for the jobs specified in **-J job_name**, **-B group_name**, or **-I application_name** parameters that match the override number specified by *overnum*. To report data for the most current override, set *overnum* to **0**.

Note: To use this parameter, you must also specify the **-J job_name**, **-B group_name**, or **-I application_name** parameter.

-p

(Optional) Generates a report that displays all the active agent plug-ins on the machine. The report only shows the data for real machines.

-R *run_num*

(Optional) Generates the report for the job run specified in *run_num*.

To specify a previous run relative to the most recent run, use the minus character (-). For example, when you specify **-r -2**, CA Workload Automation AE generates a report for the job run two runs back.

If you omit this parameter or specify 0 for the *run_num*, the report contains information about the most recent job run.

Note: This parameter is only valid with the **-s** and **-d** parameters for job reporting.

-L *print_level*

(Optional) Specifies the number of levels into the box job hierarchy about which to report. This value can be any number. For example, when you specify **-L 2**, the report contains data for the specified job (a box) and the top two levels of jobs in the box. To report on only the topmost level (that is, the box), specify **-L 0**.

Default: **-L** (report on all levels in the box)

-t

(Optional) Includes the time zone (if one is specified in the job definition) in the report. The time zone appears in parentheses beneath the job name, and the Time column of the report is based on the time zone specified in the job.

-B *group*

(Optional) Defines the group for which to generate a report.

Limits: When **-I application** is also defined, the command generates the report only for jobs that match both **-I application** and **-B group**. A null group value may be specified as **-B ""**.

Notes:

- The **-J job_name** and **-B group** or **-I application** are mutually exclusive. That is, you cannot specify the **-J job_name** with **-B group** or **-I application** and conversely.

This value allows wildcard characters.

-I *application*

(Optional) Defines the application for which to generate a report.

Limits: When **-B group** is also defined, the command generates the report only for jobs that match both **-I application** and **-B group**. A null application value may be specified as **-I ""**.

Notes:

- The **-J job_name** and **-B group** or **-I application** are mutually exclusive. That is, you cannot specify the **-J job_name** with **-B group** or **-I application** and conversely.
- This value allows wildcard characters.

-V *virtual_resource_name*

(Optional) Specifies the virtual resource definition that you want to generate a report for.

Notes:

- You can use wildcard characters.
- If you specify **-s**, the command generates a summary report that displays the resource name, the machine it is defined for (if any), the total amount, and the amount currently available.
- If specify **-d [-M *machine*]**, the command generates a summary report and detail report. The detail report displays information about the machine that is using the resource or all machines that are currently using the resource if ALL is specified in the -M option.

-X *ext_instance*

(Optional) Specifies the external instance that you want to generate a report for.

-Y *job_type*

(Optional) Defines the user-defined job type that you want to generate a report for.

Limits: Specify -Y ALL to report on all user-defined job types.

Notes:

- When -q is also defined, the command generates a query report.
- When -d option is also defined, the command generates a detailed report.

-z *globalblob_name*

(Optional) Generates a global blob report, where *globalblob_name* is the name of a global binary large object (blob).

Limits: Specify -z ALL to generate a report on all globs.

Note: You can use the percent (%) and underscore (_) characters as wildcards in this value.

-a

(Optional) Downloads the appropriate globs in text file format. This option scans and converts all new line characters in the binary large object (blob) data to conform to the text file format of the operating system. If not specified, the glob is saved without modification to a file as binary data.

Note: To use the -a option, you must also specify **-z *globalblob_name***.

-f blobfilepath

(Optional) Defines the directory to store files containing blob data. When you do not specify the directory to store blob files, CA Workload Automation AE places the blob files into the designated temporary path for the operating system by default. On Windows, the default path is determined by the TEMP environment variable, whereas on UNIX the default path is /tmp.

Note: To use the -f option, you must also specify -z *globalblob_name* or -J *job_name* with -q.

autorep Command Usage Notes

Consider the following points when using the autorep command:

Read access to objects:

- When you invoke autorep, CA Workload Automation AE verifies that you have read access to the objects (jobs, machines, global variables, and so on) you are requesting reports for. The list access for objects reported by autorep can be denied. You can only view reports for the objects in the list that you have read access to.
- There is no visual indication that a report list is incomplete. The autorep command does not display a security warning if it cannot obtain read access to one of the objects that match the input name qualifier. It also does not warn that access to view the entire list is denied.

Archived data:

Data archived with archive_events does not appear in the reports.

Exit codes:

- The exit code and output of autorep vary depending on whether a report is generated for a single object or multiple objects.
- If the name of an explicit object is specified and the object does not exist or the read access to the object is denied, autorep returns an error message and a non-zero exit code value.
- If an object name with a wildcard is specified or the ALL qualifier is used and no reports for objects can be generated, autorep returns zero as the exit code value to indicate success.

Report format:

When autorep reports nested jobs, the subordinate jobs are indented to illustrate the hierarchy.

Types of autorep Reports

You can use the autorep command to generate the following types of reports:

Job Summary

When you use the -s parameter to request a summary, autorep returns the current status (if the job is still running) or the status for a previous run. The summary report reflects the last status posted to the database.

Job Detail

When you use the -d parameter to request a detail report, autorep returns all events in the database ujo_event table for the job. If the job is running, autorep returns events for the current run. If the job is not running, it returns events for the most recent run or for the specified run.

Machine Definition and Status in the Database

When you use the -M parameter to request a machine report, autorep returns the current load, defined maximum load and factor, and current status for all machines or a specific machine defined in the database.

The valid machine statuses are:

Online

Indicates that the machine is online and available to run jobs.

Offline

Indicates that an operator used sendevent to take the machine offline.

Missing

Indicates that, during proactive monitoring of machines, the scheduler determined that the machine is not available (for example, due to a crash).

Unqualified

Indicates that the scheduler is attempting to qualify the status of an agent before switching the machine from an online to missing status.

Job Overrides

When you use the -o parameter to request an override report, autorep returns the overridden job definition fields and user, time, and run information for a specified override, based on job name and override number.

Columns in an autorep Report

The columns in an autorep report vary with the type of report requested and may include the following:

Last Start

Reports the date and time at which the specified job most recently started.

Last End

Reports the date and time at which the specified job most recently ended.

ST

Reports the completion status of the most recent run of the specified job, or, if the job is still running, the current status.

The following table lists the abbreviations used in the ST column of the autorep report, and gives the status associated with each abbreviation:

Abbreviation	Status (Event)
AC	ACTIVATED
FA	FAILURE
IN	INACTIVE
OH	ON_HOLD
OI	ON_ICE
PE	PEND_MACH
QU	QUE_WAIT
RE	RESTART
RU	RUNNING
RW	RESWAIT
ST	STARTING
SU	SUCCESS
TE	TERMINATED

Run

Reports the job run number and number of tries, separated by a slash (/).

Pri/Xit

Reports the priority/exit code associated with the specified job. If the job is in QUE_WAIT status, this column shows the priority in the queue; if the job completed, this column shows the exit code.

Status/[Event]

Reports the status for the current run of the specified job.

Time

Reports the date and time at which the scheduler or agent generated the CHANGE_STATUS event for the specified job.

Ntry

Reports the number of restart attempts for the event listed in the Status/[Event] column.

ES

Reports the processing state of the status (event).

The following table lists the abbreviations used in the ES (event state) column of the autorep report, and gives the status associated with each abbreviation:

Abbreviation	Event State
ER	Error
PD	Processed
PG	Processing
UP	Unprocessed
US	Unsent

Process Time

Reports the date and time at which the scheduler processed the CHANGE_STATUS event.

Machine

Reports the name of the machine on which the specified job ran or is running.

Max Load

Reports the max_load value defined in the specified machine's definition.

Current Load

Reports the current load on the specified machine.

Factor

Reports the factor value defined in the specified machine's definition. This is the value multiplied by a real machine's available CPU cycles to determine the relative available CPU cycles for the machine.

O/S

Reports the operating system in use on the specified machine.

Load

Reports the relative amount of processing power a job will consume on the specified machine.

Priority

Reports the queue priority of a job on the specified machine.

Status (machine reports)

Reports the current status of the machine. Options are the following:

Empty

Indicates that the machine container does not have any component machines defined.

Missing

Indicates that the machine cannot be reached.

Offline

Indicates that the machine was manually taken offline.

Online

Indicates that the machine appears to be functioning correctly.

Unqualified

Indicates that the machine state is questionable and will be re-evaluated.

Examples: Generate Reports Using autorep

The following examples generate reports using the autorep command:

Example: Generate a Summary Report

This example generates a summary report for a run of the Nightly_Download box job:

```
autorep -J Nightly_Download
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST	Run/Ntry	Pri/Xit
Nightly_Download	05/21/2010 11:27:31	05/21/2010 11:27:33	SU	197/1	0
Watch_4_file	05/21/2010 11:27:32	05/21/2010 11:27:33	SU	197/1	0
filter_data	05/21/2010 11:27:32	05/21/2010 11:27:33	SU	197/1	0
update_DBMS	05/21/2010 11:27:32	05/21/2010 11:27:33	SU	197/1	0

Example: Generate a Job Detail Report

This example generates a detail report that shows each event and status for each job:

```
autorep -J Nightly_Download -d
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST Run/Ntry	Pri/Xit
Nightly_Download	05/21/2010 11:27:31	05/21/2010 11:27:33	SU 197/1	0
Status/[Event]	Time	Ntry ES	ProcessTime	Machine
RUNNING	05/21/2010 11:27:31	1 PD	05/21/2010 11:27:31	
SUCCESS	05/21/2010 11:27:33	1 PD	05/21/2010 11:27:34	
Watch_4_file		05/21/2010 11:27:32	05/21/2010 11:27:33	SU 197/1 0
Status/[Event]	Time	Ntry ES	ProcessTime	Machine
STARTING	05/21/2010 11:27:31	1 PD	05/21/2010 11:27:31	localhost
RUNNING	05/21/2010 11:27:32	1 PD	05/21/2010 11:27:32	localhost
<Executing at WA_AGENT>				
SUCCESS	05/21/2010 11:27:33	1 PD	05/21/2010 11:27:33	localhost
filter_data		05/21/2010 11:27:32	05/21/2010 11:27:33	SU 197/1 0
Status/[Event]	Time	Ntry ES	ProcessTime	Machine
STARTING	05/21/2010 11:27:31	1 PD	05/21/2010 11:27:31	localhost
RUNNING	05/21/2010 11:27:32	1 PD	05/21/2010 11:27:32	localhost
<Executing at WA_AGENT>				
SUCCESS	05/21/2010 11:27:33	1 PD	05/21/2010 11:27:33	localhost
update_DBMS		05/21/2010 11:27:32	05/21/2010 11:27:33	SU 197/1 0
Status/[Event]	Time	Ntry ES	ProcessTime	Machine
STARTING	05/21/2010 11:27:31	1 PD	05/21/2010 11:27:31	localhost
RUNNING	05/21/2010 11:27:32	1 PD	05/21/2010 11:27:32	localhost
<Executing at WA_AGENT>				
SUCCESS	05/21/2010 11:27:33	1 PD	05/21/2010 11:27:33	localhost

In this example, Nightly_Download is a box job. Therefore, it does not enter the STARTING state, but goes directly to the RUNNING state. Box jobs do not list an associated machine, because the jobs in boxes may execute on different machines.

Example: Generate a Detailed Report for a Previous Job Run

This example generates a detailed report for the previous run of a job:

```
autorep -J RunData -d -r -1
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST Run/Ntry	Pri/Xit
RunData	05/21/2010 11:41:10	05/21/2010 11:41:10	FA 198/1	20007
Status/[Event]	Time	Ntry ES	ProcessTime	Machine
-----	-----	-----	-----	-----
STARTING	05/21/2010 11:41:09	1 PD	05/21/2010 11:41:09	localhost
RUNNING	05/21/2010 11:41:10	1 PD	05/21/2010 11:41:10	localhost
FAILURE	05/21/2010 11:41:10	1 PD	05/21/2010 11:41:10	
<Command file not found>				
[*** ALARM ***]				
JOBFAILURE	05/21/2010 11:41:10	1 PD	05/21/2010 11:41:11	localhost
[STARTJOB]	05/21/2010 11:41:41	0 PD	05/21/2010 11:41:41	

Example: Generate a Machine Summary Report

This example generates a report that lists all machines defined on the data server:

```
autorep -M ALL
```

The resulting report might resemble the following:

Machine Name	Max Load	Current Load	Factor	O/S	Status
london	100	0	1.00	Unix	Online
berlin	90	0	0.90	NT	Online
v_italy.rome	0	0	0.00	Unix	Online
v_italy.venice	0	0	0.00	Unix	Online
v_italy.venice	100	0	1.00	NT	Online
v_france.cannes	75	0	1.00	NT	Online

Example: Generate an Override Report for the Current Override

This example generates an override report that shows the current one-time override in effect for the specified job:

```
autorep -J RunData -o 0
```

The resulting report might resemble the following:

```
/* ----- over ----- */
override_job: RunData
/* Over-Ride #2 Set by User: roger@venice on [07/28/1997 16:13:59] */
/* Over-Ride CURRENTLY IN EFFECT.*/
command: /bin/rundata2
```

Example: Generate an Override Report for a Previous Override

This example generates an override report that shows the first one-time override for the specified job:

```
autorep -J RunData -o 1
```

The resulting report might resemble the following:

```
/* ----- over ----- */
override_job: RunData
/* Over-Ride #1 Set by User: roger@venice on [07/25/1997 18:23:45] */
/* Was RUN on run_num=175, Started on: 07/25 18:24:01 */
command: /bin/rundata1
```

Example: Generate a Report for Jobs in a Group

This example generates a report of information for all jobs in the HumanResources group:

```
autorep -B HumanResources
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST Run/Ntry Pri/Xit
hrbox	-----	-----	IN 0/0
hrjob1	-----	-----	IN 0/0
hrjob2	-----	-----	IN 0/0

Example: Generate a Report for Jobs in an Application

This example generates a report of information for all jobs in the YearEndHR application:

```
autorep -I YearEndHR
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST Run/Ntry Pri/Xit
yearendbox	-----	-----	IN 0/0
yearendjob1	-----	-----	IN 0/0

Example: Generate a Report for a Specific Global Variable

This example generates a report that lists the value of the global variable DAY:

```
autorep -G DAY
```

The resulting report might resemble the following:

Wednesday

Example: Generate a Report for ALL Global Variables

This example generates a report that lists the values of all global variables:

```
autorep -G ALL
```

The resulting report might resemble the following:

Global Name	Value	Last Changed
DAY	Wednesday	07/21/2005 08:15:14
AUDIT_DIR	/usr/audit	07/21/2005 08:15:12
DINNER_TIME	18:30	07/21/2005 08:15:13
MAX_VAL	2048	07/21/2005 08:15:13

Example: Generate a Report for a Specific Glob

This example generates a report that downloads the contents of the glob MYGLOB to location *download_path* as binary data:

```
autorep -z MYGLOB -f download_path
```

The resulting report might resemble the following:

Glob Name	File Name
MYGLOB	<download_path>/MYGLOB

This example generates a report that downloads the contents of the glob MYGLOB to location *download_path* as text data:

```
autorep -z MYGLOB -f download_path -a
```

The resulting report might resemble the following:

Glob Name	File Name
MYGLOB	<download_path>/MYGLOB.txt

Example: Generate a Report for ALL Globals

This example generates a report that downloads the contents of all globs to location *download_path* as binary data:

```
autorep -z ALL -f download_path
```

The resulting report might resemble the following:

Glob Name	File Name
MYGLOB	<download_path>/MYGLOB
REPORT_CHART	<download_path>/REPORT_CHART
ARCHIVED_DATA	<download_path>/ARCHIVED_DATA
JOB_SNAPSHOT	<download_path>/JOB_SNAPSHOT

This example generates a report that downloads the contents of all globs to location *download_path* as text data:

```
autorep -z ALL -f download_path -a
```

The resulting report might resemble the following:

Glob Name	File Name
MYGLOB	<download_path>/MYGLOB.txt
REPORT_CHART	<download_path>/REPORT_CHART.txt
ARCHIVED_DATA	<download_path>/ARCHIVED_DATA.txt
JOB_SNAPSHOT	<download_path>/JOB_SNAPSHOT.txt

Example: Generate a Summary Report of Jobs in a Box

This example generates a summary report about the top two levels of jobs in the box job Box3:

```
autorep -J abox -L 2
```

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST Run/Ntry Pri/Xit
abox	-----	-----	IN 0/0
abox1	-----	-----	IN 0/0
abox1job1	-----	-----	IN 0/0
abox1job2	-----	-----	IN 0/0
abox2	-----	-----	IN 0/0

Example: Generate a Detailed Report that Includes the Time Zone Specification

This example includes the time zone specification in a detailed report for the last run of the job MyJob, enter:

```
autorep -J MyJob -d -t
```

When you use the **-t** option, CA Workload Automation AE translates the time in the Time column to the time zone specified in the job definition. The time reported in the ProcessTime column is not affected by the **-t** option. It shows the time the scheduler processed the events (server time).

The resulting report might resemble the following:

Job Name	Last Start	Last End	ST	Run/Ntry	Pri/Xit
MyJob (EST5EDT)	05/21/2010 11:27:32	05/21/2010 11:27:33	SU	197/1	0
Status/[Event]	Time	Ntry	ES	ProcessTime	Machine
STARTING	05/21/2010 11:27:31	1	PD	05/21/2010 11:27:31	localhost
RUNNING	05/21/2010 11:27:32	1	PD	05/21/2010 11:27:32	localhost
<Executing at WA_AGENT>					
SUCCESS	05/21/2010 11:27:33	1	PD	05/21/2010 11:27:33	localhost

Example: Extract Job Definitions in Script Format and Output to a File

This example uses the autorep command to extract job definitions and direct the output to a file:

```
autorep -J ALL -q > dump_file
```

dump_file

Identifies the file to which to write output.

The product formats the resulting output exactly as a job definition script like the following:

```
insert_job: test_job
job_type: c
command: sleep 60
machine: juno
#owner: jerry@ca
permission: gx,ge,wx
alarm_if_fail: 1
```

Note: The owner field of each job definition is commented out (#) unless the EDIT superuser ran the autorep command to generate this report. Only the EDIT superuser can change the owner field.

If the job has one or more input blobs tied to it, in addition to the job definition, the autorep command extracts each of the job blob definitions. In this case, the product formats the resulting output like the following:

```
insert_job: test_job_with_blob
job_type: c
command: sleep 60
machine: juno
owner: jerry@juno
permission:
std_in_file: $$blobt
alarm_if_fail: 1

/* -- test_job_with_blob:insert_blob #1 -- */
insert_blob: test_job_with_blob
blob_input: <auto_blobt>multi-lined text data for job blob 1
</auto_blobt>

/* -- test_job_with_blob:insert_blob #2 -- */
insert_blob: test_job_with_blob
blob_input: <auto_blobt> multi-lined text data for job blob 2
</auto_blobt>
```

```
/* -- test_job_with_blob:insert_blob #3 -- */
insert_blob: test_job_with_blob
blob_input: <auto_blobt> multi-lined text data for job blob 3
</auto_blobt>
```

Alternatively, you can specify a location to download the blobs using the -f parameter in the following way:

```
autorep -J ALL -q -f download_path > dump_file
```

download_path

Identifies the path to download any job input blobs.

dump_file

Identifies the file to which to write output.

In this case, the product formats the resulting output like the following:

```
insert_job: test_job_with_blob    job_type: c
command: sleep 60
machine: juno
owner: jerry@juno
permission:
std_in_file: $$blobt
alarm_if_fail: 1

/* -- test_job_with_blob:insert_blob #1 -- */
insert_blob: test_job_with_blob
blob_file: <download_path>/test_job_with_blob_1.txt

/* -- test_job_with_blob:insert_blob #2 -- */
insert_blob: test_job_with_blob
blob_file: <download_path>/test_job_with_blob_2.txt

/* -- test_job_with_blob:insert_blob #3 -- */
insert_blob: test_job_with_blob
blob_file: <download_path>/test_job_with_blob_3.txt
```

You can save this file as a backup of job definitions, or you can use a text editor to quickly edit the job definitions. Use the following jil command to re-load the job definitions into the database:

```
jil < dump_file
```

Example: Generate Summary and Detail Reports for a Global Virtual Resource

This example generates a summary report and a detail report for the ren1 virtual resource. The resource is global, so it is available to all machines.

```
autorep -V ren1 -d
```

The resulting report might resemble the following:

Resource Name	Type	Machine	Defined	Available
ren1	V	---	4	1

***** Current Resource Usage *****

ResName	JobName	Run/Ntry	Status	Machine	Amount in Use
ren1	job1	1/1	RUNNING	machine1	1
ren1	job2	1/1	RUNNING	machine2	2

The -M option is not specified in the command, so the report displays all the jobs that are currently using the resource.

Example: Generate an Agent Plug-in Report

This example generates a report that displays all the active agent plug-ins on the machine named myhost.

```
autorep -M myhost -p
```

The report displays the following information:

```
machine: myhost
type: a
node_name: myhost
agent_name: WA_AGENT
Active Plug-ins:
    clearfiles
    communicationmanager
    filebrowser
    filemon
    nop
    objmon
    outbox
    refresh
    runner
    setproperty
    status
    router
```

autosec_test Command—Simulate a Call to the Security Subsystem

The autosec_test command is a client component utility used to test the security subsystem by a CA Workload Automation AE application for a given secured asset.

Use the autosec_test command to test basic communication between a CA Workload Automation AE secured application and the security subsystem and to verify enforcement of the security policies for CA Workload Automation AE assets (jobs, calendars, machines, and so on). If CA Workload Automation AE is running under the external security model, the autosec_test command returns a message indicating whether a security access check for a given resource to an operation (execute, write, read, create, or delete) passes or fails as governed by the security policy. If no *username* value is specified, the application issues a security check as the currently logged-on user.

Syntax

This attribute has the following formats:

- To test the access level of a specified asset:

```
autosec_test [CALENDAR resource | GLOBAL_VAR resource | JOB resource |  
JOBTYPE resource | MACHINE resource | RESOURCES resource]  
[C | D | R | W | X] [username]  
  
autosec_test [CYCLE resource] [C | D | R | W] [username]  
  
autosec_test [APPL resource | GROUP resource] [R | W | X] [username]  
  
autosec_test [CONTROL resource | LIST resource] [X] [username]  
  
autosec_test [OWNER resource | JOBLOG resource] [R] [username]
```

- To display the current security mode:

```
autosec_test STAT
```

- To display the version number:

```
autosec_test -X
```

- To display help for the command:

```
autosec_test -?
```

CALENDAR resource

(Optional) Issues a security check to test whether the calendar *resource* has the specified level of access (C, D, R, W, or X).

GLOBAL_VAR resource

(Optional) Issues a security check to test whether the global variable *resource* has the specified level of access (C, D, R, W, or X).

JOB resource

(Optional) Issues a security check to test whether the job *resource* has the specified level of access (C, D, R, W, or X).

JOBTYPE resource

(Optional) Issues a security check to test whether the job type *resource* has the specified level of access (C, D, R, W, or X).

MACHINE resource

(Optional) Issues a security check to test whether the machine *resource* has the specified level of access (C, D, R, W, or X).

CYCLE resource

(Optional) Issues a security check to test whether the cycle calendar *resource* has the specified level of access (C, D, R, or W).

RESOURCES resource

(Optional) Issues a security check to test whether the specified resource has the specified level of access (C, D, R, W, or X).

APPL resource

(Optional) Issues a security check to test whether the job application *resource* has the specified level of access (R, W, or X).

GROUP resource

(Optional) Issues a security check to test whether the job group *resource* has the specified level of access (R, W, or X).

JOBLOG resource

(Optional) Issues a security check to test whether the job output log file *resource* has read access.

OWNER resource

(Optional) Issues a security check to test whether the owner *resource* has read access.

C

(Optional) Tests for create access to the specified resource.

Note: This level of access is valid for CALENDAR, CYCLE, GLOBAL_VAR, JOB, MACHINE, and RESOURCES resources.

D

(Optional) Tests for delete access to the specified resource.

Note: This level of access is valid for CALENDAR, CYCLE, GLOBAL_VAR, JOB, MACHINE, and RESOURCES resources.

R

(Optional) Tests for read access to the specified resource.

Note: This level of access is valid for CALENDAR, CYCLE, GLOBAL_VAR, JOB, MACHINE, and RESOURCES resources.

W

(Optional) Tests for write access to the specified resource.

Note: This level of access is valid for CALENDAR, CYCLE, GLOBAL_VAR, JOB, MACHINE, and RESOURCES resources.

X

(Optional) Tests for execute access to the specified resource.

Limits: This level of access is valid for CALENDAR, GLOBAL_VAR, JOB, MACHINE resources, and RESOURCES resources.

username

(Optional) Issues the security check on behalf of the user *username*.

Default: If no user name is specified, access is checked on behalf of the currently logged-on user who is issuing the command.

Note: The issuing user's SERVER attribute must be set in the CA EEM environment to allow issuance of a CA EEM security check on behalf of another user.

CONTROL *resource*

(Optional) Issues a security check to test whether the administrative task control *resource* has execute access.

Limits: You can set *resource* to one of the following:

- EPLOG
- SECADM
- AUTOREP
- STOP_DEMON
- WEBADM

LIST *resource*

(Optional) Issues a security check to test whether the list *resource* has execute access.

Limits: You can set *resource* to one of the following:

- AUTOLOCAL
- AUTOCONS
- AUTOREP
- JOBDEF
- JOBDEP
- MONBRO

STAT

(Optional) Displays a message indicating the security mode that CA Workload Automation AE is running under.

Example: Determine Whether the Current User May Issue an Event for a Job

This example simulates a request by the CA Workload Automation AE secured application to the security subsystem to determine whether the current user has access to issue a STARTJOB event for the job good_test:

```
autosec_test JOB good_test X
```

If the user has execute access to the job, the following message is displayed:

Security check passed!

If the user does not have execute access to the job, the following message is displayed:

Security check failed!

Example: Determine Whether a Specified User May Issue an Event for a Job

This example simulates a request by the CA Workload Automation AE secured application to the security subsystem to determine whether the user admin@example.com can issue a STARTJOB event for the job good_test:

```
autosec_test JOB good_test X admin@example.com
```

If admin@example.com has execute access to the job, the following message appears:

```
Security check passed!
```

If admin@example.com does not have execute access to the job, the following message appears:

```
Security check failed!
```

autostatus Command—Report Job Status or Global Variable Value

The autostatus command is a client component utility that reports the current status of the specified job or the current value of a global variable to standard output. This command is especially useful in two circumstances:

- When one job needs to know the status of another.
- When complex starting conditions are required beyond those that can be easily specified in the job definition.

For example, you may need to start a job when two out of a set of three jobs have completed successfully. You could handle these dependencies through the starting conditions, but the conditions would be very cumbersome to define. Instead, you can use autostatus to check the status of the three jobs and perform the appropriate action. For more information, see the examples later in this section.

When you invoke autostatus, CA Workload Automation AE verifies that you have read access to the jobs and global variables for which you are requesting information.

Syntax

This command has the following formats:

- To report on a job:
`autostatus -J jobname [-S instance]`
- To report on a global variable:
`autostatus -G globalname [-S instance]`
- To display the version number for the command:
`autostatus -x`
- To display help for the command:
`autostatus -?`

-J *jobname*

Specifies the job to report on. The job's current status is returned to standard output.

-G *globalname*

Specifies a global variable. The variable's value is returned to standard output.

Note: The specified global variable must have been set with the sendevent command.

-S *instance*

(Optional) Defines the three-character identifier of the CA Workload Automation AE instance that the job or global variable resides on (for example, ACE).

UNIX Default: The value of \$AUTOSERV from the current environment.

Windows Default: The value of %AUTOSERV%.

Note: To use this option, you must also specify the -J or -G option.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Check the Status of a Job on the Current Instance

This example checks the current status of the job test_install on the current instance.

```
autostatus -J test_install
```

The command writes the result to standard output. For example:

```
SUCCESS
```

Example: Check the Value of a Global Variable

This example checks the value of a global variable named “Today”.

```
autostatus -G Today
```

The command writes the result to standard output. For example:

```
12/20/2011
```

Example: Replace a Condition Statement

This example uses autostatus to replace a complex condition statement. The autostatus command runs the New_Stats job when the following conditions are satisfied:

- The Account_Run, Corp_Run, and End_Run jobs ran.
- At least two of the jobs completed successfully.
- It is not the 13th day of the month.

The autostatus command is used as follows:

1. Create a job named New_Stats_Starter with the following job definition:

```
#  
# JIL for New_Stats_Starter  
#  
insert_job: New_Stats_Starter  
job_type: CMD  
machine: mombo  
command: new_stats_starter  
condition: done(Account_Run) and  
done(Corp_Run) and done(End_Run)
```

2. (UNIX only) Use the following code to create a Bourne shell script with the name new_status_starter to run for the job named New_Status_Starter. This script determines whether to start the job New_Status. If the job exits with a 0 exit code (SUCCESS), New_Status can start; if the job exits with a non-zero exit code (FAILURE), New_Status cannot start.

```
#!/bin/sh
#
# Program to determine when to start New_Status
# Check for 13th of month - if it is, exit
# with 2
if [ `date +%d` -eq 13 ]
then
exit 2
fi
# Add up the Number of SUCCESS endings
SUCCESS=0
for JobName in Account_Run Corp_Run End_Run
do
if [ `autostatus -J $JobName` = "SUCCESS" ]
then
SUCCESS='expr $SUCCESS + 1'
fi
done
if [ $SUCCESS -gt 1 ]
then
exit 0
else
exit 1
fi
```

Note: To reference jobs running on other instances, you must set **-S** instance to the three-character identifier for that instance in the call to autostatus.

3. (Windows only) Use the following code to create a batch file with the name New_Stats.bat to run for the job New_Stats_Starter. This batch file determines whether to start the job New_Stats. If the job exits with a 0 exit code (SUCCESS), New_Stats can start; if the job exits with a non-zero exit code (FAILURE), New_Stats cannot start.

```
@REM ECHO OFF
@REM Program to determine when to start New_Stats
@REM
@REM Check for 13th of month, if it is, exit with 2
@echo | more | date > testdate.out
@findstr /c:"/13/" testdate.out
@if errorlevel 1 goto not13
@if errorlevel 0 echo 13th of the month
@del testdate.out
@autostatus -J %1 > test1.dat
@findstr SUCCESS test1.dat
@if errorlevel 1 goto nosuccess
@if errorlevel 0 goto sayok
:nosuccess
@del test1.dat
@REM false is supplied in %AUTOSYS%\bin
@false 1
@goto exit:
:sayok
@del test1.dat
@goto exit
:not13
@del testdate.out
@REM false is supplied in %AUTOSYS%\bin
@false 2
:exit
```

Note: To reference jobs running on other instances, you must set –S instance to the three-character identifier for that instance in the call to autostatus.

4. Create the job New_Stats and set the following starting condition in the job definition to specify that it should start when the previous job (New_Stats_Starter) completes successfully:

```
condition: success(New_Stats_Starter)
```

autostatad Command—Report the Status of a CA AutoSys Workload Automation Adapter Job

Valid on UNIX

The autostatad command reports the current status of the specified CA AutoSys Workload Automation Adapter job to standard output. For example, you might use the autostatad command to report the status of a CA AutoSys Workload Automation for SAP job.

When CA Workload Automation AE runs an adapter job, the adapter stores the job's current status in the CA Workload Automation AE database as a global variable. The global variable name is the job name and its value is the job status and the date and time the status was written to the database.

Syntax

This command has the following format:

```
autostatad -J job_name [-h] [-s] [-t] [-x]
```

-J *job_name*

Specifies the job to report on. The job's current status is returned to standard output.

Note: You can use wildcards, including the percent (%) character, in this value.

-h

(Optional) Displays help for the command.

-s

(Optional) Displays only the status for the specified job, without the job name or any labeling. This parameter is useful for creating stripped output to include in scripts.

-t

(Optional) Displays the date and time the job's status was written to the CA Workload Automation AE database.

-x

(Optional) Writes the version number to standard output.

Notes:

- When you invoke autostataad, CA Workload Automation AE verifies that you have read access to the jobs that you are requesting information for.
- The exit code and output of autostataad vary depending on whether the status is requested for a single job or multiple jobs. If the name of an explicit job is given, and either the job does not exist or the read access to the job is denied, autostataad returns an error message and a non-zero exit code value. However, if a job name with a wildcard is specified and no status for jobs can be returned, autostataad returns zero as the exit code value to indicate success.
- If the list access for jobs reported by autostataad is denied, you can only view the status for jobs in the list to which you are granted read access. There is no visual indication that the status list is incomplete. autostataad does not display a security warning if it cannot obtain read access to one of the jobs that match the input name qualifier. It does not warn that access to view the entire list is denied.

Example: Report the Status of Similarly Named Adapter Jobs

This example returns the job name, status, and timestamp of all Adapter jobs whose names begin with autoMB.

```
autostataad -J autoMB% -t
```

The output resembles the following:

```
Job: autoMB1 Status:completed Time: 10/13/05 09:09:00  
Job: autoMB2 Status:aborted Time: 10/13/05 09:15:31  
Job: autoMB3 Status:failed Time: 10/13/04 09:09:21
```

Example: Report the Status of an Adapter Job with No Labeling

This example returns only the status of the job autoMB1 with no labeling.

```
autostataad -J autoMB1 -s
```

If appropriate, you can redirect the stripped output from this command to a file for use in a script.

autosys_secure Command—Define Security Settings

The autosys_secure command is a scheduler and client component utility that defines the EDIT superuser, EXEC superuser, database password, remote authentication method, and Windows user IDs and passwords for CA Workload Automation AE. You can also use autosys_secure to enable CA EEM security in CA Workload Automation AE and perform basic CA EEM administration.

Syntax

This command has the following formats:

- To run autosys_secure in interactive (menu-driven) mode:

```
autosys_secure
```

- To issue autosys_secure from the command line:

```
autosys_secure -u user@domain_or_host
                [-a | -c | -d | -s]
                [-editu | -execu]
                [-o old_password]
                [-p password]
                [-host domain_or_host]
                [-q]
```

- To display help for the command:

```
autosys_secure -?
```

-u *user@host_or_domain*

Defines the user whose credentials you are specifying.

Limits: Windows user names can be 1-20 characters and can contain all characters except the following:

[] + : ; “ < > . , ? / \ |

-a

(Optional) Adds the user ID specified in the **-u** parameter with the password specified in the **-p** parameter.

-c

(Optional) Changes the existing password specified in the **-o** parameter for the user specified in the **-u** parameter to the new password specified in the **-p** parameter.

-d

(Optional) Deletes the existing password specified in the **-o** parameter for the user specified in the **-u** parameter.

-s

(Optional) Lists the existing users. To specify **-s ALL**, you must also specify the **-editu**, **-execu**, or **-host** option.

- editu

(Optional) Defines the user specified by the **-u** parameter as an EDIT superuser.

- execu

(Optional) Defines the user specified by the **-u** parameter as an EXEC superuser.

-o old_password

(Optional) Defines an existing password to change (when used with the **-c** parameter) or delete (when used with the **-d** parameter).

Limits: To specify a NULL password, set this parameter to **-o**.

This parameter is required when you use the **-c** or **-d** parameters.

-p password

(Opitonal) Defines a new password.

Limits: To specify a NULL password, set this parameter to **-p**.

This parameter is required when you use the **-a** or **-c** parameters.

Windows passwords can be up to 14 characters. Passwords are case-sensitive and can contain any character except a space.

- host domain_or_host

(Optional) Changes the *host_or_domain* value for the user specified by **-u user@host_or_domain**.

Limits: To specify a NULL *host_or_domain*, set this parameter to **-host**.

This parameter is only valid when you use the **-c** parameter with the **-editu** or **-execu** parameter.

-q

(Optional) Runs autosys_secure in quiet mode so it does not display messages.

In quiet mode, you must use one of the following commands to check the return code to see whether there were errors:

- On UNIX:

```
csh echo $status  
sh or ksh echo $?
```

- On Windows:

```
echo %ERRORLEVEL%
```

When autosys_secure is successful, the return value is 0. When it is unsuccessful, the value is 1.

-?

(Optional) Displays help for the command.

Example: Add USER@HOST From the Command Line

This example adds USER@HOST from the command line.

```
autosys_secure -a -u testuser@machine1 -p testuserpass
```

Example: List all USER@HOST Defined From the Command Line

This example lists all USER@HOST defined from the command line.

```
autosys_secure -s -host ALL
```

Native and Embedded Security

CA Workload Automation AE uses native or external security for protecting jobs, machines and so on.

CA Workload Automation AE native security includes the following:

- Job-level security
- Superuser privileges
- System-level security
- UNIX and Windows file permissions

Security is initiated when either a user or the scheduler sends events that affect a job run.

CA Workload Automation AE external security is accomplished through integration with CA EEM. All GUI applications and command line interfaces have callouts to security. User-defined classes in CA EEM are used to govern what types of resources can be controlled by which users.

Note: For more information about configuring CA Workload Automation AE to work with CA EEM, see the *Security Guide*.

EDIT Superuser and EXEC Superuser

Two users have administrator privileges: the EDIT superuser and the EXEC superuser.

The EDIT superuser is the only user with permission to do the following:

- Edit or delete any job, regardless of who owns it and what permissions are set for it.
- Change the owner of a job.
- Change the database password, remote authentication method, and Windows user passwords.

The EXEC superuser is the only user with permission to do the following:

- Start or kill any job, regardless of the execution permissions on the specified job. This user can affect how jobs run, typically by issuing the sendevent command.
- Shut down the scheduler (by sending the STOP_DEMON event).

Database Password

The scheduler and command line utilities use the database password to securely access the database (event server). By default, this password is stored in the database as autosys, but the EDIT superuser can use autosys_secure to change it. The EDIT superuser must change this password before CA Workload Automation AE starts a job on a legacy agent.

Every event server in an instance must have the same database password. If you are running in dual event server mode, autosys_secure changes the password on both event servers.

Note: If CA Workload Automation AE has rolled over to single event server mode, do not change the password until you have re-established dual event server mode.

Remote Authentication Method

CA Workload Automation AE uses remote authentication methods to verify the following:

- Whether a user has permission to start a job on a client.
- Whether a scheduler has permission to start a job on a client.

The remote authentication methods are stored in the database and are referenced when a client initializes.

User and Host Management

Valid on Windows

Windows user IDs and passwords must be entered into the database for jobs to run on Windows clients. When an agent runs a job on a computer, it logs on and impersonates the user who owns the job. To enable the agent to do this, the scheduler passes both the job information and an encrypted version of the job owner's password from the database to the agent. The EDIT superuser can use autosys_secure interactively or from the command line to enter these passwords. After the EDIT superuser enters the IDs and passwords, any user who knows an existing user ID and password can change the password or delete that user ID and password.

Note: Entering passwords using autosys_secure is not related to Windows user account administration.

Encrypted Passwords

A password converted into an encrypted password is used for database connection and by CA Workload Automation AE adapters. For the database connection, the encrypted password is the password for the autosys database user. The encrypted password is stored in the config.\$AUTOSERV file on UNIX and in the registry on Windows.

Running `autosys_secure`

Before using `autosys_secure` to change the database password or the remote authentication method, you must shut down all processes that access the database (for example, application servers, schedulers, GUIs, and agents) and ensure that jobs are not running.

When `autosys_secure` is invoked in interactive mode, the menu displayed for the EDIT superuser is different from the menu displayed for users other than EDIT superuser. Only the EDIT superuser has access to all `autosys_secure` options.

Note: Any user that knows an existing user ID and password combination can change the password or delete that user ID from the database.

Running `autosys_secure` from the Command Line

You can run `autosys_secure` from the command line to enter, change, or delete Windows user IDs and passwords.

This option lets you use interactive programs, scripts, or batch files written in a scripting language (such as Perl) to enter or modify large numbers of user IDs and passwords. Perl is shipped with CA Workload Automation AE.

Enter the following command to display the usage statement for the command line options:

```
autosys_secure -h
```

Windows Agent User Authentication

When agent user authentication is enabled on a Windows computer, the agent on that computer may consult the Trusted Hosts and Trusted Users security entries from the CA Workload Automation AE Administrator to verify that the job's owner is registered on the client.

The Trusted Hosts list specifies host or domain names. All users on all hosts specified in the Trusted Hosts list are assumed to be trusted and can run jobs on the client. Only the EDIT superuser can change this list by using the CA Workload Automation AE Administrator Security window. You should use great care when changing this list because it grants access to all users on a given host.

The Trusted Users lists are owned and administered by individual users from the CA Workload Automation AE Administrator Security window on their specific clients. Trusted Users differ from Trusted Hosts in that each user on a client or domain has a personal Trusted Users list. Users can grant access to foreign users to run jobs under their current, logged-on user accounts. The Trusted Users list entries are in the *user@host_or_domain* format. All users listed in the Trusted Users field can run jobs on that client.

autosyslog Command—Display the Scheduler, Application Server, and Log File for a Job

The autosyslog command is a client component utility that displays the scheduler, application server, or agent log file for the specified job. The agent, application server, and scheduler write diagnostic messages to their respective logs as part of their normal operations and in response to detected error conditions.

The scheduler log contains a timestamped history of each event that occurs. Viewing this log is an alternative to monitoring “all jobs” and “all events.”

Because the scheduler logs all events it processes and provides a detailed trace of its activities, autosyslog provides valuable information for troubleshooting, and should always be used early in the troubleshooting process.

The autosyslog utility can be a useful diagnostic tool when jobs fail. This command, when provided with the name of a job, displays the log of the job’s most recent run.

On the agent, the agent’s log file is automatically deleted by default after a successful job run. However, the log file is not deleted at job completion if the job ended with a FAILURE status.

Syntax

This command has the following formats:

- To display the scheduler log file:
`autosyslog -e [-l num_lines] [-p regex]`
- To display the application server log file:
`autosyslog -s`
- To display a job’s log file:
`autosyslog -j job_name [-r run_num | -r run_num -n retry_num] [-tA | -tP | -t0 | -tE]`
- To display the version number of the CA Workload Automation AE instance:
`autosyslog [-x]`
- To display the command help:
`autosyslog [-?]`

-e

(Optional) Monitors the scheduler log. The last 10 lines of the scheduler log file are displayed. The log file is updated continually as processing occurs. To terminate the command, press Ctrl+C.

UNIX:

- Using autosyslog to view the scheduler log is the same as issuing the following command:
`tail -f $AUTOUSER/out/event_demon.$AUTOSERV.`
- To view the scheduler log, you must run this command on the machine that is running the scheduler or on a machine that can access the same \$AUTOUSER file system.
- The \$AUTOUSER and \$AUTOSERV environment variables must be set the same as when the scheduler was run.

Windows:

- To view the scheduler log, you must run this command on the machine that is running the scheduler or on a machine that can access the same %AUTOUSER% file system.
- The %AUTOUSER% and %AUTOSERV% environment variables must be set the same as when the scheduler was run.
- To view the scheduler log for CA Workload Automation AE installed on a Windows 2003 or earlier Windows version, you must issue the command as a user with Administrator or Power user privileges. For later versions of Windows, you can issue the command as any type of user.

-l num_lines

(Optional) Specifies the maximum number of lines in the scheduler log file to display.

-p regex

(Optional) Specifies a regular expression to search for. The command displays the lines in the scheduler log file that match the pattern.

Note: The interpretation of the regular expression depends on the operating system. The lines matched may be different across operating systems.

-S

(Optional) Monitors the application server log. To terminate the command, press Ctrl + C.

UNIX:

- To view the application server log, you must run this command on the machine that is running the application server or on a machine that can access the same \$AUTOUSER file system.
- The \$AUTOUSER and \$AUTOSERV environment variables must be set the same as when the application server was run.

Windows:

- To view the application server log, you must run this command on the machine that is running the application server or on a machine that can access the same %AUTOUSER% file system.
- The %AUTOUSER% and %AUTOSERV% environment variables must be set the same as when the application server was run.

-J *job_name*

Displays the agent log for the job *job_name*.

-r *run_num*

(Optional) Specifies the job's run number.

-n *retry_num*

(Optional) Specifies the retry number.

Default: 0 (last retry)

Limits: Positive integer

-tA | -tP | -tO | -tE

(Optional) Specifies the type of job log to display. Options are the following;

- -tA—Displays the agent log file for the job. This is the default.
- -tP—Displays the agent environment profile log file for the job.
- -tO—Displays the job's standard output file.
- -tE—Displays the job's standard error file.

Notes:

- If the -t parameter is not specified with the -j parameter, autosyslog applies the -tA specification by default.

- For legacy agent log files, when the CleanTmpFiles parameter in the configuration file is “on,” CA Workload Automation AE removes the agent log file upon successful completion of the job. In this case, the autosyslog command cannot display the log contents of jobs that completed successfully because the logs do not exist. For agent log file maintenance information, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

Example: View the Scheduler Log

This example displays the scheduler log when entered on the command line of the system where the scheduler is running, or where it ran last:

```
autosyslog -e
```

UNIX Note: You can enter this command on any machine that can access the same file system (for example, \$AUTOUSER) as the scheduler.

Windows Note: You can enter this command on any machine that can access the same file system (for example, %AUTOUSER%) as the scheduler.

Example: View the Agent Log for a Job

This example displays the agent log of the last run of the “test_install” job, when entered on the command line of the machine where the “test_install” job ran:

```
autosyslog -J test_install
```

Example: View the Log File with Profile Information on UNIX

This example displays the output file with profile information on UNIX:

```
autosyslog -j job_name -tP
```

This command displays the log file first, appending the profile output, if there is any. If no profile output file exists, the profile output section is empty.

autotimezone Command—Manage the ujo_timezones Table

The autotimezone command is a client component utility that lets you query, add, and delete entries in the ujo_timezones table. The ujo_timezones table maps cities and common aliases to valid POSIX TZ environment variables. The table contains entries for all common time zones recognized by most operating systems, and for many cities.

You can use the timezone attribute to specify entries from the ujo_timezones table in a job definition.

The ujo_timezones table has three entry types: Zone, Alias, and City, as shown in the following excerpt from the ujo_timezones table:

Entry	Type	Zone
US/Pacific	Zone	PST8PDT
US/Samoa	Alias	Pacific/Samoa
UTC	Alias	GMT+0
Vancouver	City	Canada/Pacific

All “Alias” and “City” types are eventually resolve to “Zone” types. The “Zone” types resolved to TZ variables (in the Zone column) that correspond to those recognized by the operating system for the machine on which the event server is running.

Syntax

This command has the following format:

```
autotimezone [-a alias zone] [-c city zone] [-t timezone]
              [-d alias|city] [-q alias|city|timezone|pattern]
              [-l] [-x] [-?]
```

-a alias zone

(Optional) Associates a name (*alias*) with a time zone (*zone*) and adds it to the ujo_timezones table as an alias entry.

Limits: The *alias* value can be up to 50 alphanumeric characters and can include slash (/), hyphen (-), and underscore (_) characters.

Notes:

- The *zone* value must be a time zone recognized by the operating system.
- You must separate the *alias* and *zone* values with a space.

-c city zone

(Optional) Associates a city (*city*) with a time zone (*zone*) and adds it to the ujo_timezones table as a city entry. Entries added to the table with the **-c** parameter display as type “City” in a listing of the ujo_timezones table.

Limits: The *city* value can be up to 50 alphanumeric characters and can include slash (/), hyphen (-), and underscore (_) characters.

Notes:

- The *zone* value must be a time zone recognized by the operating system.
- You must separate the *city* and *zone* values with a space.

-t timezone

(Optional) Adds a time zone entry to the ujo_timezones table.

Limits: A time zone entry must be formatted as a valid POSIX standard timezone (TZ) environment variable.

-d alias|city

(Optional) Deletes the entry associated with the specified alias or city from the ujo_timezones table.

-q alias|city|timezone|pattern

(Optional) Queries the ujo_timezones table for the settings of the specified alias, city, time zone, or pattern.

Limits: This value allows wildcard characters.

-l

(Optional) Lists all entries in the ujo_timezones table.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Add a City to the ujo_timezones Table

This example adds a city San-Jose in the US/Pacific time zone to the ujo_timezones table.

```
autotimezone -c San-Jose US/Pacific
```

Example: Delete a City from the ujo_timezones Table

This example deletes the city San-Jose from the ujo_timezones table.

```
autotimezone -d San-Jose
```

Example: Query the ujo_timezones Table

This example queries the ujo_timezones table for all entries beginning with the letter “d”.

```
autotimezone -q d%
```

The output from this command resembles the following:

Entry	Type	Zone
Dallas	City	US/Central
Denver	City	US/Mountain
Detroit	City	US/Eastern

TZ Variable Syntax

Important! The ujo_timezones table is complete and accurate and does not need modification. If you change the ujo_timezones table, you must not change or delete entries that are used by pre-existing STARTJOB and other events that were scheduled using the old ujo_timezones table.

The TZ variable has the following format:

```
std offset [dst [offset] [,start [/time] ,end [/time] ]]
```

std offset

Defines standard time for the time zone, where *offset* indicates the amount of time to add to or subtract from the local time to arrive at GMT.

Limits: This value is required.

The *offset* value has the following format:

```
[-|+]hh[:mm[:ss]]
```

-|+

(Optional) Defines whether the offset value represents a time zone west or east of GMT.

Limits: A + or no prefix indicates that the value represents a time zone west of GMT. A – prefix indicates that the value represents a time zone east of GMT.

hh

Defines the number of hours offset from GMT.

Limits: This value is required. This value can be a number in the range 0 to 24.

mm

(Optional) Defines the number of minutes offset from GMT.

Limits: This value can be a number in the range 0 to 59.

ss

(Optional) Defines the number of seconds offset from GMT.

Limits: This value can be a number in the range 0 to 59.

dst offset

(Optional) Defines daylight saving time for the time zone, where *offset* indicates the amount of time to add to the local time to arrive at GMT.

Default: When you do not specify dst, the product assumes that the time zone does not observe daylight saving time.

When you specify dst without an offset value, the product assumes that daylight saving time is one hour ahead of standard time.

Limits: The *offset* value has the following format:

`[-|+]hh[:mm[:ss]]`

`-|+`

(Optional) Defines whether the offset value represents a time zone west or east of GMT.

Limits: A + or no prefix indicates that the value represents a time zone west of GMT. A – prefix indicates that the value represents a time zone east of GMT.

hh

Defines the number of hours offset from GMT.

Limits: This value is required. This value can be a number in the range 0 to 24.

mm

(Optional) Defines the number of minutes offset from GMT.

Limits: This value can be a number in the range 0 to 59.

ss

(Optional) Defines the number of seconds offset from GMT.

Limits: This value can be a number in the range 0 to 59.

start

(Optional) Defines the day on which to change to daylight saving time from standard time.

Limits: This value can be one of the following:

jn

Defines the Julian day *n*, where *n* can be a number in the range 1 to 365. The product ignores leap days, meaning that December 31 is always day 365 and you cannot specify February 29.

n

Defines the Julian day *n*, where *n* can be a number in the range 0 to 365. In this case, the product counts leap days.

mm.n.d

Defines that the time change occurs on the *n*th occurrence of a particular day in a specific month, where:

- *mm* can be a number in the range 1 to 12. January is month 1.
- *n* can be a number in the range 0 to 5. Sunday is day 0. When *n* is 5, it means the last occurrence of the specified day in month *mm*.
- *d* can be a number in the range 0 to 6.

For example, to specify that the time change should occur on the first Sunday in April, you would specify 4.1.0.

Note: If you do not specify *start* and *end*, United States daylight saving time defaults are used. That is, *start* is the second Sunday in March and *end* is the first Sunday in November.

end

Defines the day on which to change to standard time from daylight saving time.

Limits: This value can be one of the following:

jn

Defines the Julian day *n*, where *n* can be a number in the range 1 to 365. The product ignores leap days, meaning that December 31 is always day 365 and you cannot specify February 29.

n

Defines the Julian day *n*, where *n* can be a number in the range 0 to 365. In this case, the product counts leap days.

mm.n.d

Defines that the time change occurs on the *n*th occurrence of a particular day in a specific month, where:

- *mm* can be a number in the range 1 to 12. January is month 1.
- *n* can be a number in the range 0 to 5. Sunday is day 0. When *n* is 5, it means the last occurrence of the specified day in month *mm*.
- *d* can be a number in the range 0 to 6.

For example, to specify that the time change should occur on the first Sunday in April, you would specify 4.1.0.

Note: If you do not specify *start* and *end*, United States daylight saving time defaults are used. That is, *start* is the second Sunday in March and *end* is the first Sunday in November.

time

Defines the time of day at which the change to daylight saving or standard time occurs.

Limits: This value has the following format:

hh[:mm[:ss]]

hh

Defines the hour at which the change occurs.

Limits: This value can be a number in the range 0 to 23.

This value is required.

mm

(Optional) Defines the minute at which the change occurs.

Limits: This value can be a number in the range 0 to 59.

ss

Defines the seconds at which the change occurs.

Default: 02:00:00 (2:00 a.m.)

Limits: This value can be a number in the range 0 to 59.

Note: Do not enter spaces between any of the components of the TZ variable.

UNIX Note:

For more information about the format of the TZ variable, see your system time, timezone, or environ man page.

When processing a job definition that includes a time zone, CA Workload Automation AE tries to resolve the specified time zone with the zones known to the operating system. If it is not found there, CA Workload Automation AE looks up the zone in the ujo_timezones table. If the time zone specification is not a TZ variable, the product reads the ujo_timezones table repeatedly until the specification resolves to a TZ variable. For example, assume a job definition included the following attribute:

`timezone:Berlin.`

The product resolves Berlin to Europe/Berlin the first time it reads the table. The second time the product reads the table it resolves Europe/Berlin to METS-1METD, which is a TZ variable. If a time zone is not resolved after five tries, the product considers it invalid and the job specifying the time zone fails.

Windows Note:

TZ variable syntax is compatible with Windows NT TZ variables, but has been extended to allow full control over the day and time on which daylight saving time changes occur. If the time zone specification in a job definition is not a TZ variable, the product reads the ujo_timezones table repeatedly until the specification resolves to a TZ variable. For example, assume a job definition included the following attribute:

```
timezone:Berlin
```

The product resolves Berlin to Europe/Berlin the first time it reads the table. The second time the product reads the table it resolves Europe/Berlin to MET, which is a TZ variable. If a time zone is not resolved after five tries, the product considers it invalid and the job specifying the time zone fails.

autotrack Command—Tracks Changes to the Database

The autotrack command is a client component utility that tracks changes to the database (for example, job definition changes, sendevent calls, and job overrides) and writes this information to the database.

When you query for this information, autotrack prints a report to the screen, or you can redirect the output to a file.

autotrack can track changes made to job definitions from jil, CA WCC, or the SDK. It cannot track changes made directly to the database through SQL commands.

This command is especially useful in sites that require careful monitoring of the job definition environment or where multiple users have permission to edit job definitions or send CA Workload Automation AE events.

CA Workload Automation AE stores autotrack output in two tables: ujo_audit_info and ujo_audit_msg. By default, these tables reside in the same database (or tablespace) as all other CA Workload Automation AE tables. If you frequently unload all your jobs from the database and reload, turn off autotrack while you unload and reload your jobs to prevent the database from growing unnecessarily large. After you have unloaded and reloaded your jobs, turn on autotrack.

Running archive_events helps to prevent the database from filling up with autotrack output. The archive_events *-l num_of_days* command archives information older than the specified number of days from the ujo_audit_info and ujo_audit_msg tables.

Syntax

This command has the following formats:

- To retrieve information about all events but omit the details:
autotrack
- To query changes to the database:
autotrack [-v] [-F “*from_time*”] [-T “*to_time*”]
[-U *username*] [-m *machine*] [-J *jobname*] [-t *type*]
- To update the tracking level:
autotrack -u *level*
- To display the tracking level:
autotrack -l
- To display version information:
autotrack -x

- To display help for the command:

```
autotrack -?
```

-u *level*

(Optional) Defines the level of detail that autotrack writes to the database.

Default: 0

Limits: This value can be one of the following:

0

Writes no tracking detail.

1

Tracks the following and condenses each tracked event to a one-line summary:

- Job, calendar, monitor, browser, and machine definition changes
- Job overrides
- autosys_secure, autotrack, and sendevent calls

2

Tracks the same information as level 1, but also writes the entire job definition for overrides and job definition changes. This level is very database intensive and significantly impairs JIL performance.

Note: Only the EXEC or EDIT superuser can change the tracking level.

-l

(Optional) Displays the current tracking level (0, 1, or 2).

-v

(Optional) Activates verbose autotrack reporting.

-F "from_time"

(Optional) Defines the date and time that starts the interval about which to report.

Limits: This value can be in “*MM/dd/[yy]yy hh:mm*” format, as follows:

MM

This value can be a number in the range 01 to 12 specifying the month in which to begin the reporting interval.

dd

This value can be a number in the range 01 to 31 specifying the day on which to begin the reporting interval.

yyyy

This value can be a number in the range 00 to 99 or 1900 to 2100 specifying the year in which to begin the reporting interval. When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year. If you enter 79 or less, the product precedes your entry with 20; if you enter 80 or greater, the product precedes your entry with 19.

hh

This value can be a number in the range 00 to 23 specifying the hour in which to begin the reporting interval.

mm

This value can be a number in the range 00 to 59 specifying the minute in which to begin the reporting interval.

Note: You must enclose this value in quotation marks (").

-T "to_time"

(Optional) Defines the date and time that ends the interval about which to report.

Limits: This value can be in “MM/dd/[yy]yy hh:mm” format, as follows:

MM

This value can be a number in the range 01 to 12 specifying the month in which to end the reporting interval.

dd

This value can be a number in the range 01 to 31 specifying the day on which to end the reporting interval.

yyyy

This value can be a number in the range 00 to 99 or 1900 to 2100 specifying the year in which to end the reporting interval. When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year. If you enter 79 or less, the product precedes your entry with 20; if you enter 80 or greater, the product precedes your entry with 19.

hh

This value can be a number in the range 00 to 23 specifying the hour in which to end the reporting interval.

mm

This value can be a number in the range 00 to 59 specifying the minute in which to end the reporting interval.

Note: You must enclose this value in quotation marks (").

-U *username*

(Optional) Reports changes or events initiated by the specified user.

-m *machine*

(Optional) Reports changes or events initiated from the specified machine.

-J *jobname*

(Optional) Identifies the job about which to report.

Notes:

- To report on all jobs, enter -J ALL.
- You can use the percent (%) and underscore (_) characters as wildcards in this value.

-t type

(Optional) Reports a specific event. This value can be one of the following:

A

Reports calls generated by the autosys_secure command.

B

Reports Monitor/Browser definition changes generated by jil.

C

Reports calendar definition changes generated by the autocal_asc command.

J

Reports job definition changes, sendevent -J, or overrides to a specific job generated by jil.

M

Reports machine definition changes generated by jil.

O

Reports override definition changes generated by jil.

S

Reports calls generated by the sendevent command evoked through client utilities.

T

Reports calls generated by the autotrack command (for example, changes to the tracking level).

Example: Display Level 1 Verbose Reporting About a Job

This example displays verbose reporting about the job NightlyReport.

```
autotrack -J NightlyReport -v
```

The output resembles the following:

```
jane@taurus
11/21 10:04:54
job definition change
::::::::::
jane@taurus
11/21 10:05:49
job definition change
command: date
::::::::::
jane@taurus
11/21 10:06:29
sendevent issued
```

Example: Display Level 2 Verbose Reporting About a Job

This example displays a report that includes the entire job definition with changes indicated by asterisks.

```
autotrack -J NightlyReport -v
```

The output resembles the following:

```
jane@taurus
05/26/2010 13:13:51
CAUAJM_I_50191 Job definition change.

    insert_job: NightlyReport
    job_type: CMD
    alarm_if_fail: 1
    auto_delete: -1
    auto_hold: 0
    box_terminator: 0
    command: rpt_util nightly
    date_conditions: 0
    interactive: 0
    job_load: 0
    job_terminator: 0
    machine: taurus
    max_exit_success: 0
    max_run_alarm: 0
    min_run_alarm: 0
    n_retrys: 0
    owner: "jane@taurus"
    priority: 0
    service_desk: 0
    term_run_time: 0
::::::::::
jane@taurus
05/26/2010 13:14:04
CAUAJM_I_50152 Sendevent issued.

    eoid: WCCz10000628
    job_name: env
    command: sendevent "-E" "STARTJOB" "-J" "NightlyReport"
::::::::::
jane@taurus
05/26/2010 13:23:31
CAUAJM_I_50191 Job definition change.

    insert_job: NightlyReport
    job_type: CMD
*   command: rpt_util NIGHTLY
```

Example: Report Changes After a Specific Date and Time

This example requests a report of all the changes that occurred to the job NightlyReport after 1 a.m. on November 12, 2009.

```
autotrack -F "11/12/2009 01:00" -J NightlyReport
```

Example: Report Changes Made by a Specific User

This example requests a report of all changes made by the user Sue over the weekend of May 8 and 9, 2010.

```
autotrack -U sue -F "5/8/2010 01:00" -T "5/9/2010 23:59"
```

Example: Report autosys_secure Changes from a Specific Machine

This example requests a report of all autosys_secure changes that occurred from the machine gemini.

```
autotrack -t A -m gemini
```

The output from this command resembles the following:

```
jane@gemini
11/05 19:08:12
autosys_secure change
EDIT Super-User: jane
EXEC Super-User: jane
password: *****
```

More information:

[archive_events Command—Archive Data](#) (see page 30)

chase Command—Verify Job STARTING or RUNNING Status

You can use the chase command to verify that jobs are running and to evaluate the health of the agent running a job. The chase command does the following:

1. Obtains a list of jobs from the database that have been in the STARTING state for more than 120 seconds
2. Obtains a list of jobs from the database that are in the RUNNING state
3. Passes the list of jobs that are in the RUNNING state to the associated agents

Each agent then verifies that the jobs are running and returns the process status to the chase command. If the chase command detects errors, it can be instructed to send an alarm or send an event to change the job's status to FAILURE.

You can run the chase command at regular intervals to track network problems. For example, if a computer is unreachable while running a job, the chase command detects that the computer is down and can send an alarm to alert you about the problem.

Note: The database must be available and both the scheduler and the application server must be running.

Syntax

This command has the following format:

chase [-A] [-E] [-x] [-?]

-A

(Optional) Sends an alarm when any job is stuck in the STARTING or RUNNING states.

-E

(Optional) Puts a job in FAILURE status when the job and its agent are not running on the client but the database indicates they should be. In this case, the job restarts if the job definition includes the n_retrys attribute. The scheduler must be running for chase to automatically restart jobs.

Note: When chase cannot connect to an agent computer, it cannot determine the reason. To prevent jobs from being restarted when they may already have run, chase does not change the status of jobs in this case, even when you run it with the -E parameter.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Notes:

- When chase detects errors, the client sends them to standard output. You can use the optional -A and -E parameters with chase to further define the actions to take for error conditions.
- When you run chase with no options, CA Workload Automation AE performs all chase activities and writes the results to standard output, but sends no alarms or job restart events.
- Running the chase command does not result in the automatic restart of jobs that are stuck in the STARTING state. Instead, chase writes a message to standard output that manual intervention may be required. Jobs stuck in the STARTING state should not be automatically restarted—it is possible (for example, due to network problems) that the job may already have run, and its state was not yet communicated to the database. Verify the actual status of these jobs before you manually restart them.

The following notes apply only to UNIX legacy agents (types r, L and I machines):

- For Command jobs, the legacy agent checks for the parent process ID of the UNIX process executing the command. The legacy agent also verifies that the agent has a lock on the job agent log file.
- If you disable file locking on the client, the legacy agent cannot verify that a job is running. Therefore, make sure that the directory specified by the AutoRemoteDir parameter in the configuration file is on a file system that has file locking enabled.

Example: Verify that a Job is Running, Generate an Alarm, and Restart the Job

This example verifies whether the job is running, generates an alarm if there is an inconsistency, and restarts the job if necessary.

```
chase -A -E
```

More information:

[n_retrys Attribute—Define the Number of Times to Restart a Job After a Failure](#) (see page 505)

Running chase Automatically

We recommend that you run chase automatically at regular intervals to track problems on the network. For example, if a computer becomes unreachable while running a job, chase detects that the computer is down and sends an alarm. If a user has a monitor running, they are also alerted to the problem.

On UNIX, you can run chase automatically by using CA Workload Automation AE to run it as a job. The \$AUTOSYS/install/data/chase.jil file contains the job that instructs CA Workload Automation AE to run chase every 10 minutes on the computer running the scheduler (“charley,” in the following example). You can change the specific parameters in this script to suit your own environment, and submit it to the jil command by running:

```
jil < $AUTOSYS/install/data/chase.jil
```

Example: Run chase Automatically

This example defines a job for automatically running chase:

```
# chase function
#
insert_job: chase
machine: charley
command: $AUTOSYS/bin/chase -A -E
date_conditions: yes
days_of_week: all
start_mins: 05,15,25,35,45,55
max_run_alarm: 5 /* Change if many jobs are running */
term_run_time: 10 /* Change if many jobs are running */
# These output files can be changed
std_out_file: $AUTouser/out/chase.out
std_err_file: $AUTouser/out/chase.err
```

chk_auto_up Command—Confirm Application Server, Scheduler, and Event Server Status

The chk_auto_up command is a client component that determines whether the event server (database), application server, and the scheduler are running. Because the event server and the scheduler must both be running for a job to start, the chk_auto_up command is the first utility you should run when troubleshooting why events or jobs are not processing as scheduled.

UNIX Note: chk_auto_up uses the product environment variables to locate the configuration file \$AUTOUSER/config.\$AUTOSERV, and uses the information to determine the application server, data server and database names. If chk_auto_up completes successfully, the product environment variables and the configuration file are set up correctly.

Windows Note: chk_auto_up uses the product environment variables to locate the instance configuration as defined in the CA Workload Automation AE Administrator, and uses the information to determine the application server, data server and database names. If chk_auto_up completes successfully, the product environment variables and the configuration are set up correctly.

When you issue the chk_auto_up command, the client (the machine from which you issued chk_auto_up) sends a request to the application server and waits for the application server to respond. The application server looks for schedulers who report their status as active in the database. The application server contacts each active scheduler and waits for the schedulers to respond. The application collects all results and prepares a response for chk_auto_up to report.

If you are running the instance in high availability mode with dual event servers, chk_auto_up also reports the state of the system including all schedulers and any occurrence of a database rollover or scheduler failover.

Syntax

This command has the following format:

```
chk_auto_up [-Q] [-r value] [-x] [-?]
```

-Q

(Optional) Returns the exit code without any descriptive message. This makes the command useful for inclusion in shell scripts. In this case, the return code is sufficient to indicate status. If you omit the -Q parameter, the command also returns a descriptive message.

-r value

(Optional) Checks one or more of the CA Workload Automation AE components. You can use this option to check the status of the event server and/or the application server and/or the scheduler. The value specified for the -r option determines the component to be checked.

Limits: This value can be one of the following:

001 or 1

Returns the status of the event server(s).

010 or 10

Returns the status of the scheduler(s).

011 or 11

Returns the status of the event server(s) and the scheduler(s).

100

Returns the status of the application server.

101

Returns the status of the application server and the event server(s).

110

Returns the status of the application server and the scheduler(s).

111

Returns the status of the application server, event server(s), and scheduler(s).

Note: You must use the -r option to check the status of the application server. If you invoke the chk_auto_up command without any arguments, it just checks the status of the event server(s) and scheduler(s).

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Exit Codes

The chk_auto_up command exits with two sets of exit codes, which include the following:

- Runs the application without any arguments.
- Runs the application with the –r option.

Running chk_auto_up Without Any Arguments

If you run chk_auto_up command without any arguments, the application exits with one of the following return codes:

0

Indicates that the scheduler or the event server is not running.

1

Indicates that the scheduler is not running, but the event server is running.

2

Indicates that the scheduler is not running, but the primary and dual event servers are running.

10

Indicates that the scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

11

Indicates that the scheduler and the event server are running.

12

Indicates that the scheduler and the dual event servers are running.

20

Indicates that the shadow scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

21

Indicates that the shadow scheduler and the event server are running.

22

Indicates that the shadow scheduler and the dual event servers are running.

30

Indicates that the primary and shadow schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

31

Indicates that the primary and shadow schedulers and the event server are running.

32

Indicates that the primary and shadow schedulers and the dual event servers are running.

40

Indicates that the tie-breaker scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

41

Indicates that the tie-breaker scheduler and the event server are running.

42

Indicates that the tie-breaker scheduler and the dual event servers are running.

50

Indicates that the primary scheduler and the tie-breaker scheduler are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

51

Indicates that the primary scheduler, tie-breaker scheduler, and the event server are running.

52

Indicates that the primary scheduler, tie-breaker scheduler, and the dual event servers are running.

60

Indicates that the shadow scheduler and the tie-breaker scheduler are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

61

Indicates that the shadow scheduler, tie-breaker scheduler, and the event server are running.

62

Indicates that the shadow scheduler, tie-breaker scheduler, and the dual event servers are running.

70

Indicates that the primary, shadow, and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

71

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the event server are running.

72

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the dual event servers are running.

81

Indicates that the event server is running, but the scheduler(s) are not responding.

82

Indicates that the dual event servers are running, but the scheduler(s) are not responding.

99

Indicates that one of the following has occurred:

- One or more of the following environment variables is not set or is set incorrectly:
 - AUTOSYS
 - AUTOSERV
 - AUTOUSER.
- The chk_auto_up command is issued with invalid arguments.
- The application server is not running or not configured correctly.

Running chk_auto_up with the -r Argument

If you run the chk_auto_up command with the -r argument, the application exits with any one of the following return codes.

If you run the chk_auto_up command with the -r 1 argument, it checks the event server and exits with one of the following return codes:

0

Indicates that the event server is not running.

1

Indicates that the event server is running.

2

Indicates that the dual event servers are running.

If you run the chk_auto_up command with the -r 10 argument, it checks the scheduler and exits with one of the following return codes:

0

Indicates that the scheduler is not running.

10

Indicates that the primary scheduler is running.

20

Indicates that the shadow scheduler is running.

30

Indicates that the primary and shadow schedulers are running.

40

Indicates that the tie-breaker scheduler is running.

50

Indicates that the primary and tie-breaker schedulers are running.

60

Indicates that the shadow and tie-breaker schedulers are running.

70

Indicates that the primary, shadow, and tie-breaker schedulers are running.

If you run the `chk_auto_up` command with the `-r 11` argument, it checks the scheduler and the event server, and exits with one of the following return codes:

0

Indicates that the scheduler or the event server is not running.

1

Indicates that the scheduler is not running, but the event server is running.

2

Indicates that the scheduler is not running, but the primary and dual event servers are running.

10

Indicates that the scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

11

Indicates that the scheduler and the event server are running.

12

Indicates that the scheduler and the dual event servers are running.

20

Indicates that the shadow scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

21

Indicates that the shadow scheduler and the event server are running.

22

Indicates that the shadow scheduler and the dual event servers are running.

30

Indicates that the primary and shadow schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

31

Indicates that the primary and shadow schedulers and the event server are running.

32

Indicates that the primary and shadow schedulers and the dual event servers are running.

40

Indicates that the tie-breaker scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

41

Indicates that the tie-breaker scheduler and the event server are running.

42

Indicates that the tie-breaker scheduler and the dual event servers are running.

50

Indicates that the primary and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

51

Indicates that the primary scheduler, tie-breaker scheduler, and the event server are running.

52

Indicates that the primary scheduler, tie-breaker scheduler, and the dual event servers are running.

60

Indicates that the shadow and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The `chk_auto_up` command prints a detailed error message if it is not able to connect to the event server.

61

Indicates that the shadow scheduler, tie-breaker scheduler, and the event server are running.

62

Indicates that the shadow scheduler, tie-breaker scheduler, and the dual event servers are running.

70

Indicates that the primary, shadow, and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

71

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the event server are running.

72

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the dual event servers are running.

81

Indicates that the event server is running, but the scheduler(s) are not responding.

82

Indicates that the dual event servers are running, but the scheduler(s) are not responding.

99

Indicates that one of the following has occurred:

- One or more of the following environment variables is not set or is set incorrectly:
 - AUTOSYS
 - AUTOUSER
 - AUTOUSER.
- The chk_auto_up command is issued with invalid arguments.
- The application server is not running or not configured correctly.

If you run the chk_auto_up command with the –r 100 argument, it checks the application server, and exits with one of the following return codes:

0

Indicates that the application server is not running.

100

Indicates that the application server is running.

If you run the chk_auto_up command with the –r 101 argument, it checks the application server and the event server, and exits with one of the following return codes:

0

Indicates that the event server or the application server is not running.

1

Indicates that the event server is running, but the application server is not running.

2

Indicates that the dual event server is running, but the application server is not running.

100

Indicates that the application server is running, but the event server is not running.

101

Indicates that the event server or the application server is running.

102

Indicates that the dual event server or the application server is running.

If you run the chk_auto_up command with the –r 110 argument, it checks the application server and the event server, and exits with one of the following return codes:

0

Indicates that the scheduler or the application server is not running.

10

Indicates that the primary scheduler is running, but the application server is not running.

20

Indicates that the shadow scheduler is running, but the application server is not running.

30

Indicates that the primary and shadow schedulers are running, but the application server is not running.

40

Indicates that the tie-breaker scheduler is running, but the application server is not running.

50

Indicates that the primary and tie-breaker schedulers are running, but the application server is not running.

60

Indicates that the shadow and tie-breaker schedulers are running, but the application server is not running.

70

Indicates that the primary, shadow, and tie-breaker schedulers are running, but the application server is not running.

100

Indicates that the application server is running, but the scheduler is not running.

110

Indicates that the primary scheduler or the application server is running.

120

Indicates that the shadow scheduler or the application server is running.

130

Indicates that the primary and shadow schedulers are running, or the application server is running.

140

Indicates that the tie-breaker scheduler or the application server is running.

150

Indicates that the primary and tie-breaker schedulers are running, or the application server is running.

160

Indicates that the shadow and tie-breaker schedulers are running, or the application server is running.

170

Indicates that the primary, shadow, and tie-breaker schedulers are running, or the application server is running.

If you run the chk_auto_up command with the `-r 111` argument, it checks the application server, scheduler, and the event server, and exits with one of the following return codes:

0

Indicates that the scheduler, event server, or the application server is not running.

1

Indicates that the scheduler or the application server is not running, but the event server is running.

2

Indicates that the scheduler or the application server is not running, but the primary and dual event servers are running.

10

Indicates that the scheduler is running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

11

Indicates that the scheduler and the event server are running, but the application server is not running.

12

Indicates that the scheduler and the dual event servers are running, but the application server is not running.

20

Indicates that the shadow scheduler is running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

21

Indicates that the shadow scheduler and the event server are running, but the application server is not running.

22

Indicates that the shadow scheduler and the dual event servers are running, but the application server is not running.

30

Indicates that the primary and shadow schedulers are running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

31

Indicates that the primary and shadow schedulers and the event server are running, but the application server is not running.

32

Indicates that the primary and shadow schedulers and the dual event servers are running, but the application server is not running.

40

Indicates that the tie-breaker scheduler is running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

41

Indicates that the tie-breaker scheduler and the event server are running, but the application server is not running.

42

Indicates that the tie-breaker scheduler and the dual event servers are running, but the application server is not running.

50

Indicates that the primary and tie-breaker schedulers are running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

51

Indicates that the primary scheduler, tie-breaker scheduler, and the event server are running, but the application server is not running.

52

Indicates that the primary scheduler, tie-breaker scheduler, and the dual event servers are running, but the application server is not running.

60

Indicates that the shadow and tie-breaker schedulers are running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

61

Indicates that the shadow scheduler, tie-breaker scheduler, and the event server are running, but the application server is not running.

62

Indicates that the shadow scheduler, tie-breaker scheduler, and the dual event servers are running, but the application server is not running.

70

Indicates that the primary, shadow, and the tie-breaker schedulers are running, but the event server or the application server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

71

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the event server are running, but the application server is not running.

72

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the dual event servers are running, but the application server is not running.

81

Indicates that the event server is running, but the scheduler(s) are not responding, and the application server is not running.

82

Indicates that the dual event servers are running, but the scheduler(s) are not responding, and the application server is not running.

99

Indicates that one of the following has occurred:

- One or more of the following environment variables is not set or is set incorrectly:
 - AUTOSYS
 - AUTOSERV
 - AUTOUSER.
- The chk_auto_up command is issued with invalid arguments.
- The application server is not running or not configured correctly.

100

Indicates that the scheduler or the event server is not running, but the application server is running.

101

Indicates that the scheduler is not running, but the event server or the application server is running.

102

Indicates that the scheduler is not running, but the primary and dual event servers or the application server are running.

110

Indicates that the scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

111

Indicates that the scheduler and the event server are running or the application server is running.

112

Indicates that the scheduler and the dual event servers are running or the application server is running.

120

Indicates that the shadow scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

121

Indicates that the shadow scheduler and the event server are running, or the application server is running.

122

Indicates that the shadow scheduler and the dual event servers are running, or the application server is running.

130

Indicates that the primary and shadow schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

131

Indicates that the primary and shadow schedulers and the event server are running, or the application server is running.

132

Indicates that the primary and shadow schedulers and the dual event servers are running, or the application server is running.

140

Indicates that the tie-breaker scheduler is running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

141

Indicates that the tie-breaker scheduler and the event server are running, or the application server is running.

142

Indicates that the tie-breaker scheduler and the dual event servers are running, or the application server is running.

150

Indicates that the primary and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

151

Indicates that the primary scheduler, tie-breaker scheduler, and the event server are running, or the application server is running.

152

Indicates that the primary scheduler, tie-breaker scheduler, and the dual event servers are running, or the application server is running.

160

Indicates that the shadow and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

161

Indicates that the shadow scheduler, tie-breaker scheduler, and the event server are running, or the application server is running.

162

Indicates that the shadow scheduler, tie-breaker scheduler, and the dual event servers are running, or the application server is running.

170

Indicates that the primary, shadow, and tie-breaker schedulers are running, but the event server is not running. This could happen if the database password was changed in configuration file, but the application server was not restarted. The chk_auto_up command prints a detailed error message if it is not able to connect to the event server.

171

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the event server are running, or the application server is running.

172

Indicates that the primary scheduler, shadow scheduler, tie-breaker scheduler, and the dual event servers are running, or the application server is running.

181

Indicates that the event server is running, but the scheduler(s) are not responding, or the application server is running.

182

Indicates that the dual event servers are running, but the scheduler(s) are not responding, or the application server is running.

Example: Confirm Event Server and Scheduler Status

This example determines whether the event server and scheduler are up, and displays the results on your monitor:

```
chk_auto_up
```

Example: Confirm Event Server, Application Server, and Scheduler Status

This example determines whether the event server, application server, and scheduler are up, and displays the results on your monitor:

```
chk_auto_up -r 111
```

clean_files Command—Delete Old Agent Log Files from Legacy Agents

The clean_files command is a client component utility that deletes old agent log files.

Note: This command applies to the legacy agent only (machine types r, n, L, and I). For more information about maintaining the log files and spool files for the new CA Workload Automation Agent (machine type a), see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

UNIX Notes:

- clean_files searches the database for legacy machines on which jobs were started, and then sends a command to the agents on the returned machines to purge all log files from the legacy machine's agent log directory (specified by AutoRemoteDir in the configuration file).
- Agent log files are deleted automatically only when a job finishes normally, and when the CleanTmpFiles parameter in the configuration file specifies to delete log files at the end of each job. Because agent logs for failed jobs are not deleted and can use valuable disk space, we recommend that you run clean_files daily as part of the daily DBMaint cycle.

Windows Notes:

- clean_files searches the database for legacy machines on which jobs were started, and then sends a command to the agents on the returned machines to purge all log files from each legacy machine's agent log directory (specified during installation or in the Legacy Enterprise Wide Logging Directory field on the Instance - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator).
- Agent log files are deleted automatically only when a job finishes normally, and when the Clean Temporary Files check box is selected on the Scheduler - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator. Because agent logs for failed jobs are not deleted and can use valuable disk space, we recommend that you run clean_files daily as part of the daily database maintenance cycle, which you can set up on the Scheduler - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator.

Syntax

This command has the following format:

```
clean_files -d num_days [-x] [-?]
```

-d num_days

Removes log files older than the specified number of days.

Windows Note: This option is only effective on NTFS file systems. On other file systems that do not store file timestamps (For example: FAT), this command removes all agent log files.

-x

(Optional) Displays the version number of the clean_files command.

-?

(Optional) Displays help for the command.

Note: If you experience problems running jobs successfully, the agent log files are useful diagnostic tools. In this case, do not run clean_files or remove the agent log files until the problem has been diagnosed and resolved.

Example: Delete All Agent Log Files That are More Than One Day Old

This example starts clean_files and deletes all agent log files that are more than one day old:

```
clean_files -d 1
```

cron2jil Command—Convert UNIX crontab Data to a JIL Script or Calendar File

Valid on UNIX

The cron2jil command is a client component utility that converts each line in a UNIX crontab file to a corresponding JIL script (*.jil file) and, if necessary, a run calendar file (*.cal file).

Because the cron2jil command cannot comprehensively address all job processing requirements, you should use it as a first step in converting from cron to the environment. The second step requires editing the newly created JIL and calendar files to help ensure the preferred job processing.

When cron2jil reads a crontab file, it assigns job names by combining the base name of the job's command and the line number of the file. For example, the following crontab entries would result in the job names "cp_1" and "mail_2" respectively.

```
>>0,59 0,23 * * 0,6 /bin/cp /etc/passwd /etc/passwd.bak  
>>0,59 * * * 0,6 /usr/ucb/mail root@support1 < /tmp/errorLog
```

After conversion, cron commands involving pipes and I/O redirection perform just as they did in the cron environment. If run calendars are created, cron2jil only generates calendars with one year duration. After conversion, pipe and I/O redirections may not take full advantage of the fault tolerance mechanisms of CA Workload Automation AE. For example, the exit code of a failed command in a pipe may not result in the failure of the complete command expression. Therefore, you should edit converted JIL scripts and split pipes into separate jobs with the appropriate conditions and job control. With this approach, problems can be detected and reported at the point of failure.

cron2jil does not generate JIL files for jobs that are defined in crontab to start every minute; for example, with an asterisk (*) specified in the first field of the cron listing. This special case should be remedied by defining successful job completion as a starting condition for the same job.

Note: After the *.jil or *.cal files are generated, you must use the jil or autocal_asc command to define the jobs and calendars to CA Workload Automation AE.

Syntax

This command has the following format:

```
cron2jil -f crontab_file [-d output_directory] [-i include_file] [-m machine] [-p prefix]
```

-f *crontab_file*

Defines the name of the crontab-formatted file to convert.

-d *output_directory*

(Optional) Defines the directory to which the product should write the *.jil and *.cal files.

Default: The current working directory.

-i *include_file*

(Optional) Defines a file containing JIL statements to include in every generated *.jil file. You must create this file before the conversion; it can contain any default JIL statements.

-m *machine*

(Optional) Defines the name of the machine on which the conversion should occur.

Default: localhost

-p *prefix*

(Optional) Defines a prefix to insert before each job's name. For example, if you set *prefix* to AUTO, JobA becomes AUTOJobA.

Example: Convert a crontab File to a JIL Script

This example converts the crontab file named daily:

```
cron2jil -f daily
```

dbspace Command—Calculate and Report Database Table Space

The dbspace command calculates and displays a summary of the disk space used by the CA Workload Automation AE database tables.

Syntax

This command has the following format:

`dbspace [-x] [-?]`

-x

(Optional) Returns the product version information.

-?

(Optional) Displays help for the command.

Note: If you use an Oracle database, consider the following points:

- The dbspace command may erroneously report that the database is nearly full. This can occur because dbspace calculates how much space is not allocated for extents. The dbspace command reports entire extents as used space even when they are nearly empty.
- The dbspace command may incur privilege errors during execution. Privilege errors occur because the UJOADMIN account has not been granted the ANALYZE ANY privilege or the ujoadmin role does not have SELECT privileges to DBA_TABLES. If the ujoadmin role does not have the proper authority, the following messages are displayed:

CAUAJM_E_18400 An error has occurred while interfacing with ORACLE.
CAUAJM_E_18401 Function <0exec> invoked from <execute> failed <600>
CAUAJM_E_18402 ORA-00942: table or view does not exist
CAUAJM_I_18403 Processing OCI function not used(4)
dbspace: Failed to retrieve blocksize

To grant these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba
grant ANALYZE ANY to UJOADMIN;
grant SELECT ON DBA_TABLES to UJOADMIN;
```

To revoke these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba
revoke ANALYZE ANY from UJOADMIN;
revoke SELECT ON DBA_TABLES from UJOADMIN;
```

Example: Calculate and Report Database Table Space

This example calculates and displays the disk space allocated for the CA Workload Automation AE tables:

```
dbspace
```

DBMaint Command—Maintain the CA Workload Automation AE Database

The DBMaint command runs the dbstatistics and archive_events commands to perform maintenance on the CA Workload Automation AE database.

Syntax

This command has the following format:

```
DBMaint [-x] [-?]
```

-x

(Optional) Returns version information.

-?

(Optional) Displays help for the command.

DBMaint runs the dbstatistics command to perform the following tasks:

- Updates the statistics for all the tables except the ujo_meta_enumerations, ujo_meta_properties, ujo_meta_rules, ujo_meta_types, and ujo_real_resource tables.
- Runs the dbspace command to determine the available space in the database. If the amount of free space is insufficient, the command issues warnings and generates a DB_PROBLEM alarm.

If you use an Oracle database, consider the following points:

- The DBMaint command may erroneously report that the database is nearly full. This can occur because DBMaint calculates how much space is not allocated for extents. DBMaint reports entire extents as used space even when they are nearly empty.

- The DBMaint command may incur privilege errors during the execution of either the dbstatistics or dbspace command. These privilege errors may also be incurred when running the programs standalone. Privilege errors occur because the UJOADMIN account has not been granted the ANALYZE ANY privilege or the ujoadmin role does not have SELECT privileges to DBA_TABLES.

To grant these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba
grant ANALYZE ANY to UJOADMIN;
grant SELECT ON DBA_TABLES to UJOADMIN;
```

To revoke these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba
revoke ANALYZE ANY from UJOADMIN;
revoke SELECT ON DBA_TABLES from UJOADMIN;
```

- Calculates and updates the average job run statistics in the ujo_avg_job_run table. The old data is overwritten with the new data. The command returns either a 0 or 1; 0 indicates success and 1 indicates failure.

DBMaint runs the archive_events command to remove old information from the database tables. By default, the archive_events command removes data that is more than seven days old. Specifically, archive_events removes the following:

- Events and associated alarms from the ujo_job_runs table.
- Job run information from the ujo_job_runs table.
- autotrack log information from the ujo_audit_info and ujo_audit_msg tables.

The output from DBMaint, \$AUTouser/out/DBMaint.out, reports the amount of space remaining in the database so you can monitor whether the event tables are getting full. By monitoring these values, you can calculate how many events you can safely maintain in a day before archiving.

Notes:

- If you are archiving large event tables, your database connection might time out, causing DBMaint to fail. If this occurs, edit the DBMaint script to increase the value of the -t parameter in the archive_events command. For information about editing the DBMaint script, see the *User Guide*.
- By default, the CA Workload Automation AE OS user is "autosys". However, you can define a different ID for the OS user during installation.

dbstatistics Command—Generate Data Server Statistics

The dbstatistics command is a scheduler component that generates statistics in the data servers to maintain an optimal performance environment.

Syntax

This command has the following format:

```
dbstatistics [-x] [-?]
```

-x

(Optional) Returns version information.

-?

(Optional) Displays help for the command.

The dbstatistics command performs the following tasks:

- Updates the statistics for all the tables except the ujo_meta_enumerations, ujo_meta_properties, ujo_meta_rules, ujo_meta_types, and ujo_real_resource tables.
- Runs the dbspace command to determine the available space in the database. If the amount of free space is insufficient, the command issues warnings and generates a DB_PROBLEM alarm.

If you use an Oracle database, consider the following points:

- The DBMaint command may erroneously report that the database is nearly full. This can occur because DBMaint calculates how much space is not allocated for extents. DBMaint reports entire extents as used space even when they are nearly empty.

- The DBMaint command may incur privilege errors during the execution of either the dbstatistics or dbspace command. These privilege errors may also be incurred when running the programs standalone. Privilege errors occur because the UJOADMIN account has not been granted the ANALYZE ANY privilege or the ujoadmin role does not have SELECT privileges to DBA_TABLES.

To grant these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba  
grant ANALYZE ANY to UJOADMIN;  
grant SELECT ON DBA_TABLES to UJOADMIN;
```

To revoke these privileges, run the following SQL commands:

```
sqlplus sys/<passwd>@<ORACLE_SID> as sysdba  
revoke ANALYZE ANY from UJOADMIN;  
revoke SELECT ON DBA_TABLES from UJOADMIN;
```

- Calculates and updates the average job run statistics in the ujo_avg_job_run table. The old data is overwritten with the new data. The command returns either a 0 or 1; 0 indicates success and 1 indicates failure.

Note: By default, the CA Workload Automation AE OS user is "autosys". However, you can define a different ID for the OS user during installation.

event_demon Command—Run CA Workload Automation AE

The event_demon binary (that is, the scheduler) is the program, running either as a UNIX process or a Windows service that actually runs CA Workload Automation AE.

After you start the scheduler, it continually scans the database for events to process. When it finds one, it verifies whether the event satisfies the starting conditions for any job in the database. Based on this information, the scheduler first determines what actions to take, then instructs the appropriate agent process to perform the actions. These actions may include starting or stopping jobs, checking for resources, monitoring existing jobs, or initiating corrective procedures.

eventor Command—Start the Scheduler

Valid on UNIX

The eventor command starts the scheduler (also called the event demon) on UNIX. By default, eventor runs in the background and performs the following tasks:

1. Confirms that another scheduler of the same instance is not running on the same machine (as determined by the \$AUTOSERV variable).
2. Writes scheduler log information from \$AUTOUSER/out/event_demon.\$AUTOSERV to standard output using the astail command.

Syntax

This command has the following format:

eventor [-f] [-q]

-f

(Optional) Runs the scheduler in the foreground and sends all of its output to the display where the command was issued.

Note: We do not recommend the -f option for production use. The default behavior is to run the scheduler in the background. All output is written to the \$AUTOUSER/out/event_demon.\$AUTOSERV file.

-q

(Optional) Runs eventor in quiet mode.

Notes:

- We recommend that you start the scheduler with no options set. In this situation, CA Workload Automation AE performs all restart checks, sends alarms, and records output in the log file.
- Do not try to start the scheduler by invoking the event_demon binary at the command line. You must use the eventor command to properly check and configure the scheduler environment.

Example: Start the Scheduler

This example starts the scheduler in the background and runs the chase and astail commands.

eventor

Log Files

Valid on UNIX

The eventor command starts the scheduler, which writes log information to the file named \$AUTOUSER/out/event_demon.\$AUTOSERV.

By default, eventor runs the astail -f command against the log file. This command is useful for monitoring the scheduler execution, particularly when there are problems with its startup. For example, if shared libraries required by the scheduler are missing or cannot be found, the scheduler will not start. In this case, the scheduler writes a message to its log file to indicate the problem. To exit the astail command, use Ctrl+C in the window displaying the scheduler log.

The shadow scheduler writes log information to the \$AUTOUSER/out/event_demon.shadow.\$AUTOSERV file.

The tie-breaker scheduler writes log information to the \$AUTOUSER/out/event_demon.tie-breaker.\$AUTOSERV file.

forecast Command—Report Job Flow

The forecast command reports job flows based on a date or date range. Forecast reports can help you predict what occurs when a set of conditions are predefined. You can see what happens when values are changed for each forecast period and use this information to plan workflow. Based on the current job definitions, the reported job flow displays a list of jobs predicted to run on the dates you specify.

The time frame you specify can be at most 24 hours (from midnight to midnight or a single calendar day). The forecast report displays day-based jobs and event-based jobs that are expected to occur in the specified time frame. Day-based jobs are jobs that have properties based on days. Event-based jobs are jobs that are based on days and are dependent on other events.

Syntax

This command has the following formats:

- To report a job flow for a job:

```
forecast -F "mm/dd/yyyy HH:MM" [-T "mm/dd/yyyy HH:MM"] [-J job_name] [-s] [-h]  
[-n] [-l]
```

- To report a job flow for a machine:

```
forecast -M machine_name -F "mm/dd/yyyy HH:MM" [-T "mm/dd/yyyy HH:MM"] [-s] [-h]  
[-n] [-l]
```

- To report a job flow for a job on a specific machine:

```
forecast -M machine_name -F "mm/dd/yyyy HH:MM" [-T "mm/dd/yyyy HH:MM"]  
[-J job_name] [-s] [-h] [-n] [-l]
```

- To display version information:

```
forecast -x
```

- To display the command help:

```
forecast -?
```

-J *job_name*

(Optional) Specifies the name of a job or a mask to report on. You can specify -J ALL to report on all jobs.

Default: If you do not specify the -J option, the command reports on all jobs.

Note: You can use wildcard characters.

-m *machine_name*

Specifies the name of a machine to report on. You can specify -M ALL to generate a report about all machines.

Note: This value cannot be masked.

-F "mm/dd/yyyy HH:MM"

Specifies the start date and time. The report displays workload that is expected to occur starting from this specified date and time.

-T "mm/dd/yyyy HH:MM"

(Optional) Specifies the end date and time. The report displays workload that is expected to occur up to this specified date and time.

Limits: Up to 24 hours following the start date and time

Default: If you do not specify this value, the end date and time is set to 23:59 hours following the start date and time.

-s

(Optional) Sorts the data by job names and start times.

-h

(Optional) Generates a report without header information.

-n

(Optional) Generates a report in narrow format.

-l

(Optional) Generates a report for successor jobs whose predecessors started a day before the specified time range.

-?

(Optional) Displays the help for the forecast command.

Notes:

- If the -l option is specified, the command considers all the jobs that start a day before the specified time frame. Those jobs' successors are evaluated to identify any jobs that occur within that time frame.
- To determine which day-based jobs will occur in the specified time frame, the forecast command considers the following job attributes:
 - start_mins
 - start_times
 - run_window
 - run_calendar
 - exclude_calendar
 - days_of_week

To determine which event-based jobs will occur, the command then traces the event-based jobs based on the box and child relationships, dependency relationships and average run-time information of the predecessor job.

- The forecast command interprets the logical operators specified in a condition (job dependencies) as OR. For example, consider jobs A, B, and C. Suppose that jobs A and B have start times of 10:00 and 11:00, and job C depends on the success of jobs A and B. In an AND condition, the start time of job C in its first run would be evaluated to whichever time occurs later (11:00 in this example). After both jobs A and B run to SUCCESS, job C would start at both 10:00 and 11:00 in subsequent runs. However, the forecast command interprets the condition as OR, so the generated report shows the start time for job C as both 10:00 and 11:00 for all runs.
- To project jobs during dependency traces, the forecast command uses the average run time for the job. If no average run time information is available, the command uses 1 minute.
- The forecast command ignores global variables, job overrides, cross-instance dependencies, DST changes, and the timezone attribute associated with a job. The forecast report is based on the local time zone.

Example: Create a Forecast Report for a Job

This example generates a forecast report for the testproc job. The report displays when testproc will run between 10:30 a.m. and 10:30 p.m. on December 1, 2010.

```
forecast -J testproc -F "12/01/2010 10:30" -T "12/01/2010 22:30"
```

Example: View the Duration and End Time of a Job

This example generates a forecast report for the test_4 job. The report displays when test_4 will run between 11:40 p.m. on December 14, 2009 and 23:59 hours following that time (the default end time).

```
forecast -F "12/14/2009 23:40" -n
```

The report is similar to the following:

<u>Box Name</u>	<u>Job Name</u>	<u>Machine Name</u>	<u>Start Time</u>	<u>Duration</u>	<u>End Time</u>
---	test_4	localhost	12/14/2009 23:40	1	12/14/2009 23:41

initautosys Command—Open the Instance Command Prompt Window

Valid on Windows

The CA Workload Automation AE instance command prompt lets you run client utilities, such as the jil and autorep commands. Usually, you open the instance command prompt window using the shortcut that is created when you install CA Workload Automation AE.

The initautosys command lets you open the CA Workload Automation AE instance command prompt window from the Windows Command Prompt and run a client utility. You can also specify the initautosys command in a BAT or CMD file to run a CA Workload Automation AE command from a script.

Syntax

This command has the following format:

```
initautosys -i instance -r "command_line"
```

-i *instance*

Specifies the instance name.

-r "*command_line*"

Specifies the full command line.

Notes:

- You cannot use a network drive letter. You must specify a fully qualified network name (for example, \\taurus\C\$\tmp).
- The network drive (for example, \\taurus\C\$) must be shared so that a password is not required to access it.

Note: You do not have to set the environment. The CA Workload Automation AE instance command prompt automatically sets the environment variables required to run CA Workload Automation AE commands.

Example: Open the Instance Command Prompt and Run a Command

This example opens a CA Workload Automation AE instance command prompt window on the machine where the ace instance is installed and runs the autorep command. The autorep command generates a report that lists all machines defined on the data server. You can run the intiautosys command from the Windows Command Prompt. You can also specify this command in a BAT or CMD file.

```
initautosys -i ace -r "autorep -M ALL"
```

jil Command—Run the Job Information Language Processor

The jil command is a client component utility that runs the Job Information Language (JIL) processor. You can use jil to add, update, and delete jobs, machines, monitors and browsers (reports), external instances, resources, user-defined job types, blobs, and globs. You can also use jil to insert one-time job overrides.

One advantage of using jil is that you can use available UNIX tools for file manipulation to create and control CA Workload Automation AE job definitions. For example, when you run the same command on every workstation, you can create a “generic” JIL template (text file), and copy it for each machine, changing only the machine attribute of the job. You can also iteratively copy the template to a temporary file, replace the machine name, and redirect the temporary file into the jil definition.

When you invoke the jil command, CA Workload Automation AE verifies user permissions as follows based on the subcommand or attribute used:

- Create access is required to use the insert subcommands (such as insert_job and insert_machine).
- Write access is required to use the update subcommands (such as update_job and update_machine). Write access is also required for referencing application and group names in job definitions.
- Delete access is required to use the delete subcommands (such as delete_job and delete_machine).
- Execute access is required to use the exclude_calendar, machine, owner, run_calendar, and user-defined job type attributes in job definitions.

You can use the jil command in two ways:

- Redirect a JIL script file to the jil command. This method lets you automate the maintenance of object definitions in the database. A JIL script file contains one or more subcommands (such as `insert_job`).
- Issue the jil command only and enter JIL subcommands at the prompts. This method lets you interactively maintain object definitions in the database. To exit interactive mode, enter “exit” at the prompt, or press `Ctrl+D` (UNIX only).

Syntax

This command has the following format:

```
jil [-q] [-S autoserv_instance] [-V none | job | batch | syntax] [-x] [-?]  
-q
```

(Optional) Runs the command in “quiet” mode so that it only outputs error messages. This is useful when entering a large number of jobs, so that you can easily see errors.

Default: Output error messages and status messages that indicate the success or failure of the JIL subcommands. This information is very useful and should typically be allowed to print out.

-S *autoserv_instance*

(Optional) Defines the three-character identifier of the CA Workload Automation AE instance on which the job or global variable resides (for example, ACE).

Default: If you do not specify the instance name, the command uses the value of the AUTOSERV environment variable.

-V none | job | batch | syntax | syntax

(Optional) Specifies whether the JIL processor should verify the validity of the job definitions and that jobs specified in the job dependency conditions actually exist in the CA Workload Automation AE database. This value can be one of the following:

none

Specifies that the JIL processor should perform no job dependency verification.

job

Specifies that the JIL processor should verify job dependencies on a job-by-job basis when the jobs are submitted to the database.

batch

Verifies job dependencies in the database after the JIL file is entirely processed.

syntax

Verifies job dependencies for the jobs created in this session. This is the default.

syntax

Verifies the validity of the job definition without verifying job dependencies or writing the job definitions to the database.

Note: When you set -V to job or batch, unsatisfied job dependencies (that is, jobs specified in the conditions that do not exist in the CA Workload Automation AE database) are reported to standard output. The output resembles the following:

```
Insert/Updating Job: JobA
*** WARNING: The Following Jobs are referenced in the ***
Conditions for this Job, YET are not defined!
1) JobC
Database Change WAS Successful!
```

Note: The portion of a condition statement that includes job dependencies on undefined jobs evaluates to FALSE for all conditions except not running. For example, if JobA has the condition "**success(JobB) and success(JobC)**", and JobC is not defined in the database, the condition evaluates to FALSE. If JobA depends on "**not running(jobC)**", the condition evaluates to TRUE.

Default: If you do not define the -V option, the JIL processor verifies job dependencies on a session basis when the jobs are submitted to the database.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

[Example: Redirect a File to jil](#)

This example redirects a text file named job1 containing JIL statements into the jil definition:

```
jil < job1
```

[Example: Redirect a File to jil Without Verifying Job Dependencies](#)

This example redirects a text file named job1 containing JIL statements into jil and prohibits the JIL processor from verifying the existence of jobs specified in its job dependencies:

```
jil < job1 -V none
```

More information:

[JIL Job Definitions](#) (see page 287)

JIL Syntax Rules

You must follow these rules when using JIL subcommands in a script file or at the command line:

- Subcommands have the following format:

sub_command:object_name
sub_command

Specifies one of the following JIL subcommands:

- delete_blob
- delete_box
- delete_glob
- delete_job
- delete_job_type
- delete_machine
- delete_monbro
- delete_resource
- delete_xinst
- insert_blob
- insert_glob
- insert_job
- insert_job_type
- insert_machine
- insert_monbro
- insert_resource
- insert_xinst
- override_job
- update_job
- update_job_type
- update_monbro
- update_resource
- update_xinst

object_name

Indicates the user-specified name of the object to create, update, delete, or override.

- Follow each subcommand with its corresponding attribute statements, if applicable. Attribute statements can occur in any order, and are applied to the job specified in the preceding subcommand. A subsequent subcommand begins a new set of attributes for a different job.

Attribute statements have the following format:

attribute_keyword:value

attribute_keyword

Specifies a valid JIL attribute.

value

Defines the setting to apply to the attribute.

- You can enter multiple attribute statements on the same line.
- You must separate each attribute statement with at least one space.
- You must define a box job before you can put jobs in it.
- You must define a machine to the database before you can define a job to run on it.
- When defining a virtual machine or pool, you must first define all machines it contains.
- Legal *value* settings can include upper and lowercase letters, numbers, colons (if the colon is escaped as explained in the next rule), underscores (_), and @.
- Because JIL parses subcommands based on the keyword/colon combination, you must properly escape colons used in an attribute statement's value with quotation marks ("") or a backslash (\). For example, to specify the time to start a job, specify "10:00" or 10\:00.

Windows Note: When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.

- Characters within quoted strings are not interpreted. All characters within quotes are considered as part of the string. The string will not be scanned to interpret characters that might represent something else, such as a comment.

For example:

```
description: "This description /* includes special characters */"
command: grep /* begin */ $HOME/*.log
```

- Comments in a JIL script file are interpreted using the following rules:
 - To comment an entire line, insert a pound sign (#) as the first non-blank character on a line.
 - A forward slash immediately followed by an asterisk /*) can be used to mark the start of a comment. This must be preceded by a blank or be the first non-blank character on the line. The end of the comment is identified by an asterisk immediately followed by a forward slash (*/). This must be immediately followed by a blank, except when it ends a line. There must be a corresponding end of the comment /*) delimiter for each start of the comment /*) delimiter.

This comment format syntax may span multiple lines. Following are a few examples:

```
/* this is a comment */
insert_job: job1
command: ls /*/* argument must be delimited in quotes so not to be
interpreted as a comment */
machine: localhost

insert_job: job2
command: sleep 5 /* sleep a while */
machine: localhost

insert_job: job3
/* application: accounting
box_name: master
*/
command: echo "application and box are commented out"
machine: localhost

insert_job: job4
command: rm $HOME/test/* /* delete files in my test directory */
machine: localhost
```

job_depends Command—Generate Detailed Reports of Job Dependencies and Conditions

The job_depends command is a client component that provides detailed reports about a job's dependencies and conditions. You can use this command to determine the current state of a job, its job dependencies. For a box job, you can use this command to determine the nested hierarchies specified in the definition. This command can generate a report of what jobs will run during a given interval.

When you invoke job_depends, CA Workload Automation AE verifies that you have read access to the jobs that you are requesting information for.

Syntax

This command has the following format:

```
job_depends -c | -d | -r | -t  
           [-B group]  
           [-F from_datetime] [-I application] [-J job_name]  
           [-L print_level] [-e] [-s] [-T to_datetime]  
           [-w] [-x] [-?]
```

Note: The -c, -d, -r, and -t parameters are mutually exclusive. You must specify one of these parameters.

-B group

(Optional) Defines the group to generate a report for.

Limits: When -I *application* is also defined, the command generates the report only for jobs that match both -I *application* and -B *group*. A null group value may be specified as -B "".

Note: The -J *job_name* and -B *group* are mutually exclusive. You cannot specify the -J *job_name* with -B *group*.

This value allows wildcard characters.

-c

(Optional) Returns the current state of the job and the names of any jobs that depend on it.

-d

(Optional) Returns the starting conditions for the job but includes no indication of its current status. This option displays jobs in a box hierarchically (use the -L parameter to set how many levels of nesting to display).

-r

(Optional) Returns a job's virtual resource dependencies, the amount of resource required, and whether the requirements are met. The job is specified using -J *job_name* option.

Note: If you integrate CA Workload Automation AE to work with CA Spectrum Automation Manager, this option also returns real resource dependencies, the amount of real resource required, and whether the requirements are met.

-t

(Optional) Returns the starting conditions for a job; however, the top level of jobs (or boxes) reported is limited to those that start during the interval specified by the job or box's date conditions. When a box satisfies those date conditions, all of the jobs in the box are also returned (use the -L parameter to set how many levels of nesting to display).

When you set this parameter, job_depends does not calculate all complex job dependencies when reporting on the jobs and boxes that are scheduled to run. For example, JobB may have a date condition and depend on JobA. The date conditions for JobB may be met for a given day, while those for JobA are not. As a result, JobA does not run, nor does JobB. However, JobB appears on the job_depends report. Therefore, the starting conditions are also printed. For example, the following condition is reported for JobB:

```
success(JobA)
```

-F *from_datetime*

(Optional) Defines the date and time that starts the interval about which to report.

Limits: This value can be in “*MM/dd/[yy]yy hh:mm*” format, as follows:

MM

This value can be a number in the range 01 to 12 specifying the month in which to begin the reporting interval.

dd

This value can be a number in the range 01 to 31 specifying the day on which to begin the reporting interval.

yyyy

This value can be a number in the range 00 to 99 or 1900 to 2100 specifying the year in which to begin the reporting interval. When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year. If you enter 79 or less, the product precedes your entry with 20; if you enter 80 or greater, the product precedes your entry with 19.

hh

This value can be a number in the range 00 to 23 specifying the hour in which to begin the reporting interval.

mm

This value can be a number in the range 00 to 59 specifying the minute in which to begin the reporting interval.

-I application

(Optional) Defines the application for which to generate a report.

Limits: When **-B group** is also defined, the command generates the report only for jobs that match both **-I application** and **-B group**. A null application value may be specified as **-I ""**.

Note: **-J job_name** and **-I application** are mutually exclusive. You cannot specify **-J job_name** with **-I application**.

This value allows wildcard characters.

-J job_name

(Optional) Identifies the job about which to report.

Limits: To report on all jobs, enter **-J ALL**.

Note: **-J job_name** is mutually exclusive with **-B group** and **-I application**. You cannot specify **-J job_name** with either **-B group** or **-I application**.

This value allows wildcard characters.

-L print_level

(Optional) Specifies the number of levels into the box job hierarchy about which to report. For example, when you specify **-L 2**, the report contains data for the specified job (a box) and the top two levels of jobs in the box. This value can be any number.

Default: **-L** (report about all levels in the box)

Limits: To report about only the topmost level (that is, the box); set this value to **-L 0**.

To report about all levels in the box, omit the **print_level** value.

This parameter is valid only when you define the **-d** or **-t** parameters.

-e

Generates all start times between the dates and times specified in the -F and -T options.

Limits: This parameter is only valid with the -t parameter.

-s

(Optional) Sorts the jobs in an order based on their next start time.

Limits: This parameter is only valid with the -t parameter.

-T *to_datetime*

(Optional) Defines the date and time that ends the interval about which to report.

Limits: This value can be in “MM/dd/[yy]yy hh:mm” format, as follows:

MM

This value can be a number in the range 01 to 12 specifying the month in which to end the reporting interval.

dd

This value can be a number in the range 01 to 31 specifying the day on which to end the reporting interval.

yyyy

This value can be a number in the range 00 to 99 or 1900 to 2100 specifying the year in which to end the reporting interval. When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year. If you enter 79 or less, the product precedes your entry with 20; if you enter 80 or greater, the product precedes your entry with 19.

hh

This value can be a number in the range 00 to 23 specifying the hour in which to end the reporting interval.

mm

This value can be a number in the range 00 to 59 specifying the minute in which to end the reporting interval.

-w

(Optional) Displays all output columns in widened format.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Report About Jobs Scheduled to Run on a Specific Day

This example lists the jobs that are scheduled to run on Christmas day, 2007.

```
job_depends -t -J ALL -F "12/25/2007 00:00" -T "12/26/2007 00:00"
```

Example: Report the Current Condition Status of a Specific Job

This example generates a report about the current condition status of the JobX job.

```
job_depends -c -J JobX
```

monbro Command—Run a Monitor or Report

The monbro command is a client component that runs a monitor or report (browser) that has already been defined and returns the results to standard output.

When you configure a monitor with sound on, the monitor uses the sound capabilities of the machine on which it is running.

UNIX Note: The sound clips must be pre-recorded, and the machine running monbro must have access to the \$AUTOUSER/sounds directory.

The monitor or report definition must reside on the database for the instance you are accessing. The monbro command can connect to any database that the instance is configured to use.

When you invoke monbro, CA Workload Automation AE verifies that you have read access to the jobs you are attempting to monitor.

Syntax

This command has the following format:

```
monbro -N name [-P poll_frequency] [-q] [-x] [-?]
```

-N *name*

Defines the monitor or report to run.

Limits: To run all monitors or reports, specify **-N ALL**.

You can use the percent (%) character as a wildcard in this value.

-P *poll_frequency*

(Optional) For monitors only, *poll_frequency* defines the interval (in seconds) at which to poll the database.

Default: 10

-q

(Optional) Displays monbro definition in JIL format.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Run a Monitor

This example runs the monitor Mon1 (which is defined in the default database):

```
monbro -N mon1
```

Example: Run a Monitor and Display Monitor Definitions

This example runs the monitor Mon1 (which is defined in the default database) and displays the monitor definitions:

```
monbro -N mon1 -q
```

The monitor definitions returned resemble the following:

```
insert_monbro: xxx
mode: m
all_events: Y
job_filter: a
sound: N
alarm_verif: N
insert_monbro: xxx2
mode: b
all_events: N
alarm: Y
all_status: N
running: N
success: Y
failure: Y
terminated: N
starting: N
restart: N
job_filter: b
job_name: box
currun: N
after_time: "11/11/1997 12:12"
```

sendevent Command—Send Events

The sendevent command is a client component utility that sends events to start or stop jobs, stop the scheduler, put a job on hold, set global variables, bring a real machine online or offline, cancel a scheduled event, and so on.

Issuing the sendevent command is the only method of externally sending most events. You can also use sendevent to communicate with any instance, provided the machine on which the command runs can connect with the application server associated with that instance.

CA Workload Automation AE writes the sent event to the database. The scheduler continually polls the database and reads and processes events on receipt.

Note: To issue a sendevent command for a job, you must have execute permission for that job. Only alarms, comments, and set global can be sent without required permissions.

When you invoke sendevent, CA Workload Automation AE verifies user permissions as follows based on the command parameters:

- Create access is required to add a new global variable.
- Write access is required to change the value of an existing global variable or to send the CHANGE_PRIORITY event.
- Execute access is required to send the following events: CHANGE_STATUS, COMMENT, FORCE_STARTJOB, JOB_OFF_HOLD, JOB_OFF_ICE, JOB_ON_HOLD, JOB_ON_ICE, KILLJOB, RELEASE_RESOURCE, REPLY_RESPONSE, RESTARTJOB, SEND_SIGNAL, SET_GLOBAL, STARTJOB, and STOP_DEMON.
- If you use the -B *group* or -I *application* parameters, execute access to as-control SENDEVENT_GRPAPP is required.
- Delete access is required to delete a global variable or to send the DELETEJOB event.
- Machine write access is required to send a MACH_OFFLINE or MACH_ONLINE event. In default CA Workload Automation AE security mode, the user must be an EXEC superuser.

Syntax

This command has the following formats:

- To start a job, delete a job, put a job on hold or take it off hold, or to put a job on ice or take it off ice:

```
sendevent {-E DELETEJOB | -E FORCE_STARTJOB | -E STARTJOB | -E RESTARTJOB |
           -E JOB_ON_HOLD | -E JOB_OFF_HOLD | -E JOB_ON_ICE | -E JOB_OFF_ICE}
           {-J job_name | -B group | -I application | -B group -I application}
           [-S autoserv_instance] [-P priority]
           [-M max_send_trys] [-C comment]
```

- To change the status of a job:

```
sendevent -E CHANGE_STATUS -s status [-S autoserv_instance]
           {-J job_name | -B group | -I application | -B group -I application}
           [-P priority] [-M max_send_trys]
           [-q job_queue_priority] [-C comment]
```

- To send an alarm:

```
sendevent -E ALARM -A alarm [-S autoserv_instance]
           [-P priority] [-M max_send_trys] [-C comment]
```

- To send a SEND_SIGNAL or KILLJOB event:

```
sendevent {-E SEND_SIGNAL | -E KILLJOB} [-S autoserv_instance]
           {-J job_name | -B group | -I application | -B group -I application}
           [-k signal_name|signal_numbers]
           [-P priority] [-M max_send_trys] [-C comment]
```

- To define a global variable:

```
sendevent -E SET_GLOBAL -G "global_name=value" [-S autoserv_instance]
           [-P priority] [-M max_send_trys] [-C comment]
```

- To change the queue priority for a job:

```
sendevent -E CHANGE_PRIORITY -q job_queue_priority [-S autoserv_instance]
           {-J job_name | -B group | -I application | -B group -I application}
           [-P priority] [-M max_send_trys] [-C comment]
```

- To cancel unprocessed events:

```
sendevent -E event -U [-T "time_of_event"] [-S autoserv_instance]
           [-P priority] [-M max_send_trys] [-C comment]
```

- To send a response to a job that requires manual intervention:

```
sendevent -E REPLY_RESPONSE -J job_name -r response
           [-S autoserv_instance] [-P priority]
           [-M max_send_trys] [-C comment]
```

- To place a real machine online or take it offline:

```
sendevent {-E MACH_ONLINE | -E MACH_OFFLINE} -N machine  
          [-P priority] [-M max_send_trys] [-C comment]
```

- To send a STOP_DEMON event:

```
sendevent -E STOP_DEMON [-S autoserv_instance]  
          [-P priority] [-M max_send_trys] [-C comment]
```

- To specify a file that contains the sendevent commands for batch processing:

```
sendevent -F file
```

- To specify a stand-alone comment that is posted to the event log:

```
sendevent -E COMMENT -c comment
```

- To display the version number for the command:

```
sendevent -x
```

- To display help for the command:

```
sendevent -?
```

-E *event*

Specifies one of the following events to send:

- [ALARM](#) (see page 896)
- [CHANGE_PRIORITY](#) (see page 897)
- [CHANGE_STATUS](#) (see page 897)
- [COMMENT](#) (see page 899)
- [DELETEJOB](#) (see page 899)
- [FORCE_STARTJOB](#) (see page 900)
- [JOB_OFF_HOLD](#) (see page 900)
- [JOB_OFF_ICE](#) (see page 901)
- [JOB_ON_HOLD](#) (see page 901)
- [JOB_ON_ICE](#) (see page 901)
- [KILLJOB](#) (see page 902)
- [MACH_OFFLINE](#) (see page 903)
- [MACH_ONLINE](#) (see page 903)

- [RELEASE_RESOURCE](#) (see page 904)
- [REPLY_RESPONSE](#) (see page 904)
- [RESTARTJOB](#) (see page 904)
- [SEND_SIGNAL](#) (see page 905)
- [SET_GLOBAL](#) (see page 904)
- [STARTJOB](#) (see page 905)
- [STOP_DEMON](#) (see page 905)

-S *autoserv_instance*

(Optional) Defines the three-character identifier of the CA Workload Automation AE instance that the job or global variable resides on (for example, ACE).

Default: If you do not specify the instance name, the command uses the value of the AUTOSERV environment variable.

-A *alarm*

(Optional) Defines the name of the alarm to send.

Note: This parameter is required when you specify **-E ALARM**. This parameter does not apply to the other **-E** options.

-J *job_name*

(Optional) Defines the job to send the specified event to.

Limit: Up to 64 alphanumeric characters

Note: This parameter is mutually exclusive with the **-B group** and **-I application** parameters. For all events except STOP_DEMON, COMMENT, ALARM, MACH_OFFLINE, MACH_ONLINE and SET_GLOBAL, *one* of the following parameters are required:

- **-J *job_name***
- **-B *group***
- **-I *application***
- **-B *group* -I *application***

-s status

(Optional) Specifies the status to change to. Options are the following:

- FAILURE
- INACTIVE
- RUNNING
- STARTING
- SUCCESS
- TERMINATED

The status of the jobs specified by the **-J job_name**, **-B group**, or **-I application** parameters are changed to this value.

Notes:

- This parameter is required when you specify **-E CHANGE_STATUS**. This parameter does not apply to the other **-E** options.
- Changing the status to RUNNING does not cause the job to run. It only changes the status of the job in the database.
- Changing the status of a box to INACTIVE also changes all the jobs in the box to INACTIVE.
- If you change the status of a RUNNING job to FAILURE, SUCCESS, or TERMINATED (completion states), the job is rescheduled to run at the next start time.
- If you change the status of a RUNNING job to INACTIVE, RUNNING, or STARTING (states that are not terminal), the job is not rescheduled.

-P priority

(Optional) Assigns a priority to the event. Assign a high priority if the event must be processed immediately (for example, to put a job that is about to start in ON_HOLD status).

Default: 10

Limits: This value can be a number in the range 1 to 1000, where 1 is the highest priority and 1000 is the lowest.

-M max_send_trys

(Optional) Defines the maximum number of times that sendevent attempts to send the event to the database. You can specify any number of attempts.

Default: 0 (sendevent tries to send the event indefinitely.)

Note: If all the specified send attempts fail, sendevent finishes with an exit code of 1.

-q *job_queue_priority*

(Optional) Defines the new queue priority to assign to the job.

Limits: This value can be a number in the range 0 to 99, where 0 is the highest priority (indicating to start the job immediately) and 99 is the lowest.

Note: This parameter is required when you specify **-E CHANGE_PRIORITY**. This parameter does not apply to the other **-E** options.

-G "global_name=value"

(Optional) Defines the name and value of the global variable that you want to define. Use this parameter with **-E SET_GLOBAL**.

Limits:

- *global_name* can be up to 30 characters and the *value* can be up to 100 characters.
- *global_name* can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (_), pound (#), hyphen (-), and space ().
- Leading and trailing spaces in *value* are discarded.

Notes:

- To delete a global variable from the database, set it with a value of DELETE. For example:
`sendevent -E SET_GLOBAL -G "global_name=DELETE"`
- When you set a global variable, jobs can use the syntax for the indicated job attributes in the following table to reference the variable at runtime:

Job Attribute	Global Variable Syntax
condition	VALUE <i>global_name</i> operator “ <i>value</i> ”
command	\$\$ <i>global_name</i> or \$\$\{ <i>global_name</i> \}
std_*_file	\$\$ <i>global_name</i> or \$\$\{ <i>global_name</i> \}
watch_file	\$\$ <i>global_name</i> or \$\$\{ <i>global_name</i> \}

- Global variables are stored in the database; they are not set in the environment. Therefore, you cannot reference them in the job's default or user-defined profile.

-C *comment*

(Optional) Specifies a comment to attach to the event. This comment is only used as documentation. For example, you can use this parameter to document why you sent a KILLJOB event. The comment is attached to the KILLJOB event and is viewable in an autorep report.

Limits: Up to 255 alphanumeric characters, entered as a single line

Notes:

- When you use this parameter with **-E COMMENT**, it specifies a stand-alone comment that is posted to the event log.
- Enclose values that contain spaces in quotation marks (""). For example:
-C "This is a comment"

-U

(Optional) Cancels the event specified by **-E *event*** when the event has not yet processed.

Notes:

- When multiple future events exist, you can use **-U** with **-T "time_of_event"** to cancel a specific event.
- If you do not define **-T "time_of_event"**, all events that satisfy the specified parameters are cancelled.
- The canceled events are not deleted from the database. The **-U** parameter changes the status.

-T "time_of_event"

Specifies the date and time at which to process the event. Specify the value in "MM/dd/[yy]yy hh:mm" format, as follows:

MM

Specifies the month in which to process the event.

Limits: 01-12

dd

Specifies the day on which to process the event.

Limits: 01-31

[yy]yy

Specifies the year in which to process the event.

Limits: 00-99 or 1900-2100

Notes:

- When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year.
- If you enter 79 or less, the value is preceded by 20 (for example, 2012). If you enter 80 or greater, the value is preceded by 19 (for example, 1992).

hh

Specifies the hour in which to process the event.

Limits: 00-23

mm

Specifies the minute in which to process the event.

Limits: 00-59

Default: Process the event immediately.

-k *signal_name|signal_numbers*

UNIX Notes:

- For processes running on UNIX, -k *signal_numbers* specifies the signal numbers that the application is watching. The agent sends the first signal, sleeps for five seconds, then sends the next signal, and so on.
- This parameter is required for the SEND_SIGNAL event.
- -k *signal_numbers* is only valid with -E SEND_SIGNAL or -E KILLJOB. This parameter does not apply to the other -E options.
- This value can contain a comma-delimited list of signals to send to the process.
- For KILLJOB events, this parameter overrides any KillSignals specified in the configuration file.
- To send signals to an entire process group, precede the appropriate *signal_numbers* values with a minus sign (-).

Windows Notes:

- For processes running on Windows, -k *signal_name* specifies the name of the semaphore that the application is watching.
- -k *signal_name* is valid only with **-E SEND_SIGNAL**. This parameter does not apply to the other **-E** options.
- Windows does not support the concept of process groups. When you issue a KILLJOB event for a job that runs an executable (*.exe), KILLJOB kills the process specified in the command definition. When you issue a KILLJOB event for a job that runs something other than an *.exe (for example, *.bat, *.cmd, or *.com), KILLJOB terminates only the CMD.EXE process that CA Workload Automation AE used to launch the job. The Job Status is set according to the return code of the killed CMD.EXE process and can be one of the following: SUCCESS, FAILURE, or TERMINATED. Processes launched by user applications or batch (*.bat) files are not killed. To work around this limitation, you can modify your programs to watch for a signal from a CA Workload Automation AE job running on a Windows machine, and you can implement this parameter for the SEND_SIGNAL event.

-B group

(Optional) Defines the group to run the command for. You can use wildcard characters.

Notes:

- If **-I application** is also specified, the command affects only items that match both **-I application** and **-B group**.
- This parameter is mutually exclusive with the **-J job_name** parameter.

-I application

(Optional) Defines the application for which to run the command. You can use wildcard characters.

Notes:

- If **-B group** is also specified, the command affects only items that match both **-I application** and **-B group**.
- This parameter is mutually exclusive with the **-J job_name** parameter.

-N machine

(Optional) Identifies the target real machine for the MACH_OFFLINE and MACH_ONLINE events.

Limits: Up to 80 alphanumeric characters

Note: You cannot issue a MACH_ONLINE or MACH_OFFLINE event for a virtual machine.

-r *response*

(Optional) Specifies the user response to send to the job that requires manual intervention. The job resumes running after it receives the user response. Use this parameter with **-E REPLY_RESPONSE**.

Notes:

- You must also specify the **-J** option.
- Do not use the **-r** option with the **-B** and **-G** options. Otherwise, you will get an error.
- When you use the **-r** option, the **-C** option is ignored.

-F *file*

Specifies the name of a file that contains the sendevent commands for batch processing. Batch processing helps to improve the client utility performance because the program initialization happens only once for all the sendevent commands included in the file.

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Example: Manually Start a Job

This example starts the job `test_install`, which has no starting conditions (and therefore must be started manually):

```
sendevent -J test_install -E STARTJOB
```

Example: Force a Job to Start

This example forces the job `wait_job` (which is waiting on the completion of another job) to start and explains the reason for doing so:

```
sendevent -J wait_job -E FORCE_STARTJOB -C "tired of waiting, forced it"
```

Example: Place a Job On Hold

This example changes the status of the job `ready_to_run` to `ON_HOLD` to prevent its execution, and assigns the sendevent command a high priority so it will be sent immediately:

```
sendevent -J ready_to_run -E JOB_ON_HOLD -P 1
```

Example: Take a Job Off Hold

This example changes the status of the job ready_to_run from ON_HOLD to OFF_HOLD so that the job can run:

```
sendevent -J ready_to_run -E JOB_OFF_HOLD
```

Example: Prevent a Job from Running During a Specific Interval

This example puts the job named lock_out on and off hold to prevent it from running between the hours of 11:00 a.m. and 2:00 p.m.

This command puts the job on hold at 11:00 a.m.:

```
sendevent -J lock_out -E JOB_ON_HOLD -T "11/08/2007 11:00"
```

This command takes the job off hold at 2:00 p.m.:

```
sendevent -J lock_out -E JOB_OFF_HOLD -T "11/08/2007 14:00"
```

Example: Write a Comment into the Scheduler Log File

This example writes a comment into the scheduler log file:

```
sendevent -E COMMENT -C "have not received EOD files - an hour late again"
```

Example: Stop the Scheduler at a Specific Time

This example stops the scheduler at 2:30 a.m. on November 9, 2007 and includes a comment explaining why:

```
sendevent -E STOP_DEMON -T "11/09/2007 02:30" -C "stopped for upgrade"
```

Example: Change the Run Priority of a Job

This example changes the job resource_hog (which is currently at priority 1) to a lower priority, and issues the sendevent command up to five times:

```
sendevent -J resource_hog -E CHANGE_PRIORITY -q 10 -M 5
```

This command only changes the job queue priority for the next run of the job.

Example: Kill a Job Running on Another Instance

This example kills the job wrong_job, which is running on another instance (PRD):

```
sendevent -J wrong_job -E KILLJOB -S PRD
```

Example: Define a Global Variable

This example sets the global variable "today" to a value of "12/25/2007":

```
sendevent -E SET_GLOBAL -G "today=12/25/2007"
```

Example: Delete a Global Variable

This example deletes the global variable "today":

```
sendevent -E SET_GLOBAL -G "today=DELETE"
```

Example: Cancel all Unprocessed Events for a Job

This example cancels all unprocessed JOB_OFF_HOLD events for the job RunData:

```
sendevent -E JOB_OFF_HOLD -J RunData -u
```

Example: Take a Machine Offline

This example takes the machine mozart offline:

```
sendevent -E MACH_OFFLINE -N mozart
```

Example: Place a Machine Online

This example puts the machine mozart online:

```
sendevent -E MACH_ONLINE -N mozart
```

Example: Send a UNIX Signal to a Job

This example sends the UNIX signal number 1 to the job RunData:

```
sendevent -E SEND_SIGNAL -J RunData -k 1
```

Example: Watch for a Signal When Running a Job on Windows

When running a job on Windows, you can program your application to watch for a signal. More than one job can be programmed to watch for the same signal. This example uses a thread to create and then block on the semaphore MyFirstSignal. You can then signal this semaphore using the sendevent SEND_SIGNAL command. Then, as shown in the example program, the watchsem job watches for a signal on both the MyFirstSignal and MySecondSignal semaphores. A second application, such as a child process of the original job (application), could also watch and react to these signals. To signal the jobs, issue the following sendevent command from the Instance Command Prompt window:

```
sendevent -E SEND_SIGNAL -J watchsem -K MyFirstSignal
```

The example program, which you can compile using cl watchsem.c is as follows:

```
#include <windows.h>
/*Prototypes*/
void WatchForAutosysSignal(char *SignalName);
DWORD WINAPI WatchAutosysSignalThread(LPVOID Parm);
void main(void)
{
    printf("This is your application running.\n");
    printf("Call the WatchForAutosysSignal function\
with a name of a semaphore.\n");
    WatchForAutosysSignal("MyFirstSignal");
    printf("You can set up multiple watches if you choose.\n");
    WatchForAutosysSignal("MySecondSignal");
    printf("Go on about your application.");
    Sleep(-1);
}
void WatchForAutosysSignal(char *SignalName)
{
    HANDLE ThisThread;
    DWORD tid;
    char *SignalCopy;
    /* Use this to pass in the signal name without referencing any */
    /* global values */
    SignalCopy=_strupr(SignalName);
    ThisThread=CreateThread(NULL,0,WatchAutosysSignalThread,
    (LPVOID)SignalCopy,0,&tid);
    if (ThisThread==NULL)
```

```
{  
    free(SignalCopy);  
    printf("Failed to CreateThread for WatchAutosysSignalThread\\n"  
          "error %ld.\n",GetLastError());  
    return;  
}  
printf("Created thread for WatchAutosysSignalThread.\n");  
if (!CloseHandle(ThisThread)) /*Drop the handle*/  
printf("Failed to Close the thread handle error\\n"  
      "%ld.\n",GetLastError());  
return;  
}  
DWORD WINAPI WatchAutosysSignalThread(LPVOID Parm)  
{  
    HANDLE ThisSem;  
    DWORD dwrc;  
    char *SignalName;  
    SignalName=(char *)Parm;  
    printf("Beginning to wait for %s.\n",SignalName);  
    /* Autosys will Open the event semaphore and then call*/  
    /* PulseEvent to signal the semaphore and reset it to */  
    /* unsignaled. This way you can watch for the signal */  
    /* multiple times. If more than one application is waiting */  
    /* for the signal, make sure to create it with the */  
    /* ManualReset value set to TRUE. */  
    ThisSem=CreateEvent(NULL,TRUE,FALSE,SignalName);  
    if (ThisSem==NULL)  
    {  
        printf("Failed to create semaphore %s with error\\n"  
              "%ld.\n",SignalName,GetLastError());  
        return(0);  
    }  
    dwrc=WaitForSingleObject(ThisSem,INFINITE);  
    if (dwrc!=WAIT_OBJECT_0)  
    {  
        printf("WaitForSingleObject for %s failed with %d and\\n"  
              "%d.\n",SignalName,dwrc,GetLastError());  
        CloseHandle(ThisSem);  
        return(0);  
    }  
    printf("Received signal on %s.\n",SignalName);  
    CloseHandle(ThisSem);  
    printf("You may react to the signal in any way you choose.\n");  
    exit(0); /* In this case just kill the application */  
}
```

Example: Send a User Response to a Job

This example sends a user response to the overwrite_file job. The job is in WAIT_REPLY state until it receives the user response. After the job receives the user response, the job continues to run.

```
sendevent -J overwrite_file -E REPLY_RESPONSE -r YES
```

Example: Restart a Failed z/OS Job

Suppose that a z/OS job named ZTEST has five steps in the JCL. When the job runs, it fails at STEP2 because the program called at that step has an error. After you fix the error, you can manually restart the job at STEP2 instead of running all the steps.

The following command manually restarts ZTEST job at STEP2:

```
sendevent -E RESTARTJOB -J ZTEST -o STEP2
```

Example: Pass a File to the sendevent Command

Suppose that you create a file named /tmp/send10Kevents.txt that contains the following sendevent commands:

```
sendevent -E CHANGE_STATUS -j job1 -s INACTIVE  
sendevent -E CHANGE_STATUS -j job2 -s INACTIVE  
sendevent -E CHANGE_STATUS -j job3 -s INACTIVE  
....  
sendevent -E CHANGE_STATUS -j job10000 -s INACTIVE
```

To run all those commands, you can pass the to the sendevent command as follows:

```
sendevent -f /tmp/send10Kevents.txt
```

time0 Command—Calculate Internal CA Workload Automation AE Time

The time0 command is a client component that calculates the internal CA Workload Automation AE time and inserts it in MM/DD/YYYY hh:mm:ss format.

Syntax

This command has the following format:

```
time0 [-A autotime | -T "datetimestamp"] [-S] [-O "dateformat"] [-x] [-?]
```

-A *autotime*

(Optional) Converts the internal CA Workload Automation AE date and time information to MM/DD/YYYY hh:mm:ss format.

Limits: You cannot specify **-A autotime** if you specify **-T datetimestamp**.

-T *datetimestamp*

(Optional) Identifies a date and time to convert to the internal CA Workload Automation AE format.

Limits: You cannot specify **-T datetimestamp** if you specify **-A autotime**.

-S

(Optional) Returns the internal CA Workload Automation AE time format.

-O *dateformat*

(Optional) Specifies a format with which to override the MM/DD/YYYY hh:mm:ss format.

Limits: You must enclose the dateformat value in quotation marks ("").

-x

(Optional) Returns the command version information to standard output.

-?

(Optional) Displays help for the command.

Note: When you use the time0 command with no arguments, the client returns the current CA Workload Automation AE time and date information.

Example: Return the Current CA Workload Automation AE Date and Time Information

This example returns the current CA Workload Automation AE time and date information:

```
time0
```

The return from the command resembles the following:

```
CAUAJM I 50096 Current AutoTime(internal): 1121265931
```

Example: Convert the CA Workload Automation AE Date and Time to MM/DD/YYYY hh:mm:ss Format

This example converts the CA Workload Automation AE date and time to MM/DD/YYYY hh:mm:ss format:

```
time0 -A 1121265931
```

The return from the command resembles the following:

```
CAUAJM I 50097 External Time: 07/13/2005 07:45:31
```


Chapter 3: Job Types

This chapter describes each job type that you can define. Each job description includes a summary of the job's required and optional JIL attributes.

Job Types

Job types define the type of work to be scheduled. For example, you can create a CMD job to run a Windows command file, an FTP job to download a file from a server, or an SAPEM job to monitor for the triggering of an SAP event.

When you create a job definition, you can specify the job type. If you do not specify a job type in the definition, the job type is set to CMD by default. Each job type has required and optional attributes that define the job.

Box Jobs

A Box job (or box) is a container for other jobs. You can define a Box job to organize and control process flow. The box itself performs no actions, but it can trigger other jobs to run. Boxes can contain other boxes that contain jobs related by starting conditions or other criteria. Boxes let you group and manage jobs that have similar starting conditions or complex logic flows.

Required Attributes

To define a Box job, you must specify the following attribute:

- [job_type: BOX](#) (see page 464)

Optional Attributes

You can specify the following optional attributes for a Box job:

- [alarm_if_fail](#) (see page 295)
- [application](#) (see page 296)
- [auto_delete](#) (see page 304)
- [auto_hold](#) (see page 305)
- [avg_runtime](#) (see page 306)
- [box_failure](#) (see page 314)
- [box_name](#) (see page 316)
- [box_success](#) (see page 317)
- [box_terminator](#) (see page 319)
- [condition](#) (see page 335)
- [date_conditions](#) (see page 362)
- [days_of_week](#) (see page 363)
- [description](#) (see page 366)
- [exclude_calendar](#) (see page 380)
- [group](#) (see page 409)
- [job_terminator](#) (see page 461)
- [max_run_alarm](#) (see page 476)
- [min_run_alarm](#) (see page 484)
- [must_complete_times](#) (see page 493)
- [must_start_times](#) (see page 499)
- [n_retrys](#) (see page 505)
- [notification_id](#) (see page 509)
- [notification_msg](#) (see page 510)
- [owner](#) (see page 538)
- [permission](#) (see page 543)
- [priority](#) (see page 551)
- [run_calendar](#) (see page 606)
- [run_window](#) (see page 607)
- [send_notification](#) (see page 664)
- [service_desk](#) (see page 665)

- [start_mins](#) (see page 676)
- [start_times](#) (see page 677)
- [svcdesk_attr](#) (see page 698)
- [svcdesk_desc](#) (see page 700)
- [svcdesk_imp](#) (see page 701)
- [svcdesk_pri](#) (see page 702)
- [svcdesk_sev](#) (see page 703)
- [term_run_time](#) (see page 706)
- [timezone](#) (see page 722)

Example: Define a Box Job

This example defines a box job named EOD_box that depends on the EOD_watch job. The box job runs only if EOD_watch completes with a SUCCESS status.

```
insert_job: EOD_box
job_type: BOX
condition: success(EOD_watch)
```

Command Jobs

You can define a Command (CMD) job to schedule workload to run on a Windows or UNIX computer. On Windows, the CMD job runs a Windows command file. On UNIX, the CMD job runs a script or executes a command.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows.

Required Attributes

To define a Command job, you must specify the following attributes:

- [job_type: CMD](#) (see page 464)
- [command](#) (see page 323)
- [machine](#) (see page 473)

Optional Attributes

You can specify the following optional attributes for a Command job:

- [blob_file](#) (see page 312)
- [blob_input](#) (see page 313)
- [chk_files](#) (see page 320)
- [envvars](#) (see page 378)
- [fail_codes](#) (see page 381)
- [heartbeat_interval](#) (see page 410)
- [interactive](#) (see page 433)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [max_exit_success](#) (see page 475)
- [profile](#) (see page 557)
- [shell](#) (see page 668)
- [std_err_file](#) (see page 679)
- [std_in_file](#) (see page 683)
- [std_out_file](#) (see page 687)
- [success_codes](#) (see page 692)
- [ulimit](#) (see page 729)

Notes:

- All UNIX jobs that run a command (a binary file) require a user ID that has the authority to run the job on the agent computer. The [owner attribute](#) (see page 538) in the job definition specifies this user ID (the default is the user who invokes jil to define the job).
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a UNIX Command

This example runs the /bin/touch command on the file named /tmp/test_run.out. The job runs on the UNIX client computer named unixagent.

```
insert_job: test_run
job_type: CMD /* This attribute is optional for Command jobs. CMD is the default. */
machine: unixagent
command: /bin/touch /tmp/test_run.out
```

Example: Run a Windows Command

This example runs the c:\bin\test.bat command on the Windows client computer named winagent. The command is enclosed in quotation marks because the path contains a colon.

```
insert_job: test_run
job_type: CMD /* This attribute is optional for Command jobs. CMD is the default. */
machine: winagent
command: "c:\bin\test.bat"
```

CPU Monitoring Jobs

You can define a CPU Monitoring (OMCPU) job to monitor the CPU usage of the computer where the specified agent is installed.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a CPU Monitoring job, you must specify the following attributes:

- [job_type: OMCPU](#) (see page 464)
- [machine](#) (see page 473)

Optional Attributes

You can specify the following optional attributes for a CPU Monitoring job:

- [cpu_usage](#) (see page 360)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [lower_boundary](#) (see page 470)
- [monitor_mode](#) (see page 488)
- [no_change](#) (see page 507)
- [poll_interval](#) (see page 547)
- [inside_range](#) (see page 431)
- [upper_boundary](#) (see page 734)

Notes:

- If you specify WAIT or CONTINUOUS in the monitor_mode attribute, you must also specify the lower_boundary attribute, the upper_boundary attribute, or both.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor Available CPU

This example continuously monitors the CPU available on the unixagent computer. The job polls the CPU usage every 60 seconds. Each time the available CPU is within 75 and 95 percent, an alert is written to the scheduler log file. The job continues monitoring the CPU usage until it is ended manually.

```
insert_job: omcpu_job
job_type: OMCPU
machine: unixagent
cpu_usage: FREE
inside_range: TRUE
poll_interval: 60
monitor_mode: CONTINUOUS
lower_boundary: 75
upper_boundary: 95
```

Database Monitor Jobs

You can define a Database Monitor (DBMON) job to monitor a database table for an increase or decrease in the number of rows. To monitor the database table for specific changes, you can add a monitor condition to the job definition. When the condition is met, the job completes.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Attributes

To define a Database Monitor job, you must specify the following attributes:

- [job_type: DBMON](#) (see page 464)
- [machine](#) (see page 473)
- [tablename](#) (see page 704)

Optional Attributes

You can specify the following optional attributes for a Database Monitor job:

- [connect_string](#) (see page 349)
- [continuous](#) (see page 353)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [monitor_cond](#) (see page 487)
- [monitor_type](#) (see page 492)
- [user_role](#) (see page 738)

Notes:

- All database jobs require a database resource location. You must define this value; otherwise, the job fails. Your agent administrator can specify a default location using the db.default.url parameter on the agent, or you can specify the connect_string attribute in the job definition.
- All database jobs require a user ID that has the appropriate permissions to access the information in the database. The job runs under that user ID. The [owner_attribute](#) (see page 538) in the job definition specifies the user ID (the default is the user who invokes jil to define the job).
- Windows authentication is not supported.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor for an Increase in the Number of Rows in a Table Using a Condition

This example monitors the Inventory_List table for an increase in the number of rows. If the number of rows increases and the number of units of ProductA has fallen below 1000, the job completes. The database user ID is set to the user who invokes jil to define the job (the default owner).

```
insert_job: dbmon_job
job_type: DBMON
machine: dbagt
tablename: Inventory_List
monitor_type: INCREASE
monitor_cond: ProductA < 1000
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

Database Stored Procedure Jobs

You can define a Database Stored Procedure (DBPROC) job to invoke a procedure stored in a database. You can add criteria to the job definition to test the procedure's output. If the result matches the criteria, the job completes successfully. When the procedure runs, the output parameter values from the database are returned to CA Workload Automation AE. You can view the output parameter values in the job's spool file. By default, the agent separates the output parameter values in the return string with a vertical bar (|).

If you use an Oracle or SQL Server database, you can also define a Database Stored Procedure job to run a stored function.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Attributes

To define a Database Stored Procedure job, you must specify the following attributes:

- [job_type: DBPROC](#) (see page 464)
- [machine](#) (see page 473)
- [sp_name](#) (see page 673)

Optional Attributes

You can specify the following optional attributes for a Database Stored Procedure job:

- [connect_string](#) (see page 349)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [result_type](#) (see page 600)
- [sp_arg](#) (see page 670)
- [success_criteria](#) (see page 693)
- [user_role](#) (see page 738)

Notes:

- All database jobs require a database resource location. You must define this value; otherwise, the job fails. Your agent administrator can specify a default location using the db.default.url parameter on the agent, or you can specify the connect_string attribute in the job definition.
- All database jobs require a user ID that has the appropriate permissions to access the information in the database. The job runs under that user ID. The [owner_attribute](#) (see page 538) in the job definition specifies the user ID (the default is the user who invokes jil to define the job).
- Windows authentication is not supported.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Invoke a Stored Procedure from a SQL Server Database

This example invokes the byroyalty stored procedure located in the pubs database. When the job runs, a value of 40 is passed to the input parameter named percentage.

```
insert_job: sp1_job
job_type: DBPROC
machine: DB_agent
owner: sa
sp_name: byroyalty
sp_arg: ignore=yes, name=percentage, argtype=IN, datatype=INTEGER, value=40
connect_string:"jdbc:sqlserver://myhost:1433;DatabaseName=pubs"
```

Database Trigger Jobs

You can define a Database Trigger (DBTRIG) job to monitor a database table for added, deleted, or updated rows. To monitor the database table for specific changes, you can add a condition to the job definition. When the condition is met, the job completes.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Attributes

To define a Database Trigger job, you must specify the following attributes:

- [job_type: DBTRIG](#) (see page 464)
- [machine](#) (see page 473)
- [dbtype](#) (see page 365)
- [tablename](#) (see page 704)

Optional Attributes

You can specify the following optional attributes for a Database Trigger job:

- [connect_string](#) (see page 349)
- [continuous](#) (see page 353)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [trigger_cond](#) (see page 724)
- [trigger_type](#) (see page 726)
- [user_role](#) (see page 738)

Notes:

- All database jobs require a database resource location. You must define this value; otherwise, the job fails. Your agent administrator can specify a default location using the db.default.url parameter on the agent, or you can specify the connect_string attribute in the job definition.
- All database jobs require a user ID. This user ID must be authorized to create triggers on the database or schema the table belongs to. For Microsoft SQL Server, this user ID must own the database table identified by the tablename attribute. The job runs under the user ID. The owner attribute in the job definition specifies the user ID (the default is the user who invokes jil to define the job).

- Windows authentication is not supported.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Specify a Trigger Type for SQL Server

This example monitors the stores table for an added row or a deleted row. The job runs under the sa user, who owns the table and is authorized to create triggers on the database or schema the table belongs to. The job remains in a RUNNING state waiting for an added or deleted row. When a row is either added or deleted, the job completes.

```
insert_job: dbtrig1
job_type: DBTRIG
machine: DB_agent
trigger_type: DELETE,INSERT
tablename: stores
dbtype: MSSQL
owner: sa@myhost
connect_string:"jdbc:sqlserver://myhost:1433;DatabaseName=pubs"
```

Disk Monitoring Jobs

On UNIX and Windows systems, you can define a Disk Monitoring (OMD) job to monitor the available or used space on a disk or logical partition. On i5/OS systems, you can define a Disk Monitoring job to monitor storage space in the file systems mounted on the i5/OS operating system.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a Disk Monitoring job, you must specify the following attributes:

- [job_type: OMD](#) (see page 464)
- [machine](#) (see page 473)
- [disk_drive](#) (see page 371)

Note: If you specify WAIT or CONTINUOUS in the monitor_mode attribute, you must also specify the lower_boundary attribute, the upper_boundary attribute, or both.

Optional Attributes

You can specify the following optional attributes for a Disk Monitoring job:

- [disk_format](#) (see page 372)
- [disk_space](#) (see page 374)
- [inside_range](#)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [lower_boundary](#) (see page 470)
- [monitor_mode](#) (see page 488)
- [no_change](#) (see page 507)
- [poll_interval](#) (see page 547)
- [upper_boundary](#) (see page 734)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Continuously Monitor Used Space on a UNIX Partition

This example continuously monitors used disk space on a local UNIX partition. Each time the used disk space falls between 90 and 100 percent, an alert is written to the scheduler log file. The job continues monitoring the disk space until it is ended manually.

```
insert_job: unix_used
job_type: OMD
machine: unixagent
disk_drive: /export/home
disk_space: USED
disk_format: PERCENT
monitor_mode: CONTINUOUS
lower_boundary: 90
```

Entity Bean Jobs

You can define an Entity Bean (ENTYBEAN) job to create an entity bean, update the property values of an existing entity bean, or remove an entity bean from the database.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define an Entity Bean job, you must specify the following attributes:

- [job_type: ENTYBEAN](#) (see page 464)
- [machine](#) (see page 473)
- [bean_name](#) (see page 311)
- [initial_context_factory](#) (see page 429)
- [operation_type](#) (see page 512)
- [provider_url](#) (see page 560)

In addition, if the operation_type is UPDATE, you must also specify the following attributes:

- [finder_name](#) (see page 386)
- [finder_parameter](#) (see page 387)
- [method_name](#) (see page 482)

If the operation_type is REMOVE, you must also specify the following attributes:

- [finder_name](#) (see page 386)
- [finder_parameter](#) (see page 387)

Optional Attributes

You can specify the following optional attributes for an Entity Bean job:

- [j2ee_user](#) (see page 452)
- [job_class](#) (see page 458)

In addition, if the operation_type is CREATE, you can also specify the following optional attributes for an Entity Bean job:

- [create_name](#) (see page 358)
- [create_parameter](#) (see page 359)

If the operation_type is UPDATE, you can also specify the following optional attribute for an Entity Bean job:

- [modify_parameter](#) (see page 485)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Create an Entity Bean

Suppose that you want to create an entity bean that stores information about a customer such as the customer ID and phone number. The initial context factory supplied by the JNDI service provider is `weblogic.jndi.WLInitialContextFactory`. The service provider's URL is `t3://localhost:7001`, where `localhost` is the domain name of the WebLogic application server and `7001` is the ORB port. When the job runs, the entity bean instance is created.

```
insert_job: create
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
create_name: createcustomer
operation_type: CREATE
create_parameter: String="customerid", String="800-555-0100"
```

Example: Update an Entity Bean

Suppose that you want to update the phone number for the Acme company to `800-555-0199`. The customer entity bean stores the customer ID and phone number. The primary key for the customer is the customer ID. To find the entity bean, the job uses the Acme's customer ID. When the job runs, the Acme company's phone number is changed.

```
insert_job: update
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: UPDATE
method_name: changephone
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
modify_parameter: String="800-555-0199"
```

Example: Remove an Entity Bean

Suppose that you want to remove the customer record for the Acme customer. The record is stored in the database by the customer ID. When the job runs, the row in the customer table that corresponds to the Acme customer ID is removed.

```
insert_job: remove
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: REMOVE
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
```

File Trigger Jobs

You can define a File Trigger (FT) job to monitor file activity for UNIX, Windows, or i5/OS systems. The File Trigger job can monitor when a file is created, updated, deleted, expanded, or shrunk, and when a file exists or does not exist.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a File Trigger job, you must specify the following attributes:

- [job_type: FT](#) (see page 464)
- [machine](#) (see page 473)
- [watch_file](#) (see page 740)

Optional Attributes

You can specify the following optional attributes for a File Trigger job:

- [continuous](#) (see page 353)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [watch_file_change_type](#) (see page 746)
- [watch_file_change_value](#) (see page 748)
- [watch_file_groupname](#) (see page 750)

- [watch_file_owner](#) (see page 752)
- [watch_file_recursive](#) (see page 754)
- [watch_file_type](#) (see page 756)
- [watch_file_win_user](#) (see page 769)
- [watch_no_change](#) (see page 772)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor for an Increase in File Size to a Certain Amount

This example monitors the test file in the data directory. When that file size reaches 100 bytes or more, the job completes.

```
insert_job: ft_unix_size
job_type: FT
machine: unixagt
watch_file: /data/test
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: 100
```

File Watcher Jobs

You can define a File Watcher (FW) job to monitor for the existence and size of a file. CA Workload Automation AE considers the watched file complete when the file reaches the minimum file size specified in the `watch_file_min_size` attribute and the file reaches a "steady state" during the polling interval. "Steady state" indicates that the watched file has not grown during the specified interval.

Note: To run these jobs, your system requires one of the following:

- CA WA Agent for UNIX, Linux, Windows, or i5/OS
- Legacy agent for Unicenter AutoSys Job Management r4.5.1 through r11

Required Attributes

To define a File Watcher job, you must specify the following attributes:

- [job_type: FW](#) (see page 464)
- [machine](#) (see page 473)
- [watch_file](#) (see page 740)

Optional Attributes

You can specify the following optional attributes for a File Watcher job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [watch_file_min_size](#) (see page 751)
- [profile](#) (see page 557)
- [watch_interval](#) (see page 770)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the `agentparm.txt` file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor a File Every 120 Seconds

This example monitors the watch_file.log file on the winagent computer. The job completes when the file reaches 10,000 bytes and maintains a steady state for at least 120 seconds (30 seconds for the agent's global poll interval plus 90 seconds for the watch_interval).

```
insert_job: fw_job
job_type: FW
machine: winagent
watch_file: "c:\tmp\watch_file.log"
watch_file_min_size: 10000
watch_interval: 90
```

Example: Monitor a File Every 60 Seconds on a Legacy Agent

This example monitors the watch_file.log file on the unixagent computer. The unixagent is a legacy agent. The job completes when the file reaches 10000 bytes and maintains a steady state for 60 seconds.

```
insert_job: fw_job
job_type: FW
machine: unixagent
watch_file: /tmp/watch_file.log
watch_file_min_size: 10000
watch_interval: 60
```

FTP Jobs

You can define an FTP job to automate File Transfer Protocol (FTP) transfers. The FTP job can download a file from a remote FTP server to your agent computer or upload a file from your agent computer to a remote FTP server. The FTP job can use an existing FTP server or another agent as an FTP server. The FTP job always acts as an FTP client.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define an FTP job, you must specify the following attributes:

- [job_type: FTP](#) (see page 464)
- [machine](#) (see page 473)
- [ftp_local_name](#) (see page 392)
- [ftp_remote_name](#) (see page 396)
- [ftp_server_name](#) (see page 400)

Optional Attributes

You can specify the following optional attributes for an FTP job:

- [ftp_command](#) (see page 389)
- [ftp_compression](#) (see page 390)
- [ftp_local_user](#) (see page 395)
- [ftp_server_port](#) (see page 402)
- [ftp_transfer_direction](#) (see page 403)
- [ftp_transfer_type](#) (see page 404)
- [ftp_use_SSL](#) (see page 406)
- [job_class](#) (see page 458)

Notes:

- All FTP jobs require an FTP user ID. The [owner attribute](#) (see page 538) in the job definition specifies this user ID (the default is the user who invokes jil to define the job).
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Upload a File to a Remote Windows Computer

In this example, a file named textfile is uploaded from a local UNIX computer and copied to a remote Windows computer. Both locations include a complete path in the ftp_remote_name and ftp_local_name attributes. The remote Windows computer uses an agent as the FTP server. Its IP address is 172.16.0.0.

```
insert_job: FTP_REMOTE
job_type: FTP
machine: unixagent
ftp_server_name: 172.16.0.0
ftp_server_port: 123
ftp_transfer_direction: UPLOAD
ftp_transfer_type: A
ftp_remote_name: "/C:/Temp/textfile"
ftp_local_name: /export/home/ftpfiles/ftpdata/textfile
owner: test@172.16.0.0
```

Note: To transfer a file from a Windows computer, substitute a Windows path in the ftp_local_name attribute and enclose the entire path in quotation marks.

HTTP Jobs

You can define an HTTP job to invoke a program over HTTP.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define an HTTP job, you must specify the following attributes:

- [job_type: HTTP](#) (see page 464)
- [machine](#) (see page 473)
- [provider_url](#) (see page 563)

Optional Attributes

You can specify the following optional attributes for an HTTP job:

- [filter](#) (see page 383)
- [invocation_type](#) (see page 434)
- [j2ee_authentication_order](#) (see page 439)
- [j2ee_conn_domain](#) (see page 440)
- [j2ee_conn_origin](#) (see page 442)
- [j2ee_no_global_proxy_defaults](#) (see page 443)
- [j2ee_parameter](#) (see page 444)
- [j2ee_proxy_domain](#) (see page 446)
- [j2ee_proxy_host](#) (see page 447)
- [j2ee_proxy_origin_host](#) (see page 449)
- [j2ee_proxy_port](#) (see page 450)
- [j2ee_proxy_user](#) (see page 451)
- [job_class](#) (see page 458)
- [method_name](#) (see page 482)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an HTTP Job to Perform a Google Search

Suppose that you want to define a job to perform a Google search and have the results returned to the job's spool file. You also want to refine your search results by specifying a filter. In this example, the job uses the HTTP GET method to perform the Google search on "ca workload automation". When the job runs, the job's spool file includes all matches that contain the filter AE.

```
insert_job: google
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://google.com/search"
j2ee_authentication_order: BASIC,DIGEST,NTLM
filter: .*AE.*
j2ee_parameter: q="ca workload automation"\
```

Example: Define an HTTP Job to Subscribe to a Mailing List

Suppose that you want to define a job to subscribe to a mailing list located on a local server. You want to add the email address test@abc.com to the list. The servlet path is /examples/servlets/servlet/TheServlet.

```
insert_job: subscribe
job_type: HTTP
machine: appagent
invocation_type: POST
provider_url: "http://localhost:8080"
method_name: /examples/servlets/servlet/TheServlet
j2ee_parameter: key1="subscribe", key2="test@abc.com"
```

Example: Define an HTTP Job to Perform an HTTP Query

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

i5/OS Jobs

You can define an i5/OS job to schedule workload to run on an i5/OS system. The job can run a program or an i5/OS command. You can run i5/OS jobs in the root file system, open systems file system (QOpenSys), and library file system (QSYS).

Note: To run these jobs, your system requires CA WA Agent for i5/OS.

Required Attributes

To define an i5/OS job, you must specify the following attributes:

- [job_type: I5](#) (see page 464)
- [machine](#) (see page 473)
- [i5_name](#) (see page 423)

Optional Attributes

You can specify the following optional attributes for an i5/OS job:

- [fail_codes](#) (see page 381)
- [i5_action](#) (see page 411)
- [i5_cc_exit](#) (see page 413)
- [i5_curr_lib](#) (see page 415)
- [i5_job_desc](#) (see page 416)
- [i5_job_name](#) (see page 417)
- [i5_job_queue](#) (see page 418)
- [i5_lda](#) (see page 419)
- [i5_lib](#) (see page 420)
- [i5_library_list](#) (see page 421)
- [i5_others](#) (see page 426)
- [i5_params](#) (see page 427)
- [i5_process_priority](#) (see page 428)
- [job_class](#) (see page 458)
- [max_exit_success](#) (see page 475)
- [success_codes](#) (see page 692)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a CL Program

This example runs a program named MFGDATA on an i5/OS system. The MFGDATA program is contained in the library named PRODJOBS.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: MFGDATA
i5_lib: PRODJOBS
```

IP Monitoring Jobs

You can define an IP Monitoring (OMIP) job to monitor an IP address or a port on an IP address. An IP Monitoring job monitors for a running or stopped status. The job can return the result immediately or wait for the specified device to reach the specified state.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

To monitor remote IP addresses through the agent, the agent must run as root (on the CA WA Agent for UNIX, Linux, or Windows) or under a profile with sufficient authority to use the system ping command (on the CA WA Agent for i5/OS). If it runs as a user without root privileges, the job will show complete but with the message "Ping (*ip address*) insufficient privilege" in the status field and transmitter.log.

Required Attributes

To define an IP Monitoring job, you must specify the following attributes:

- [job_type: OMIP](#) (see page 464)
- [machine](#) (see page 473)
- [ip_host](#) (see page 435)

Optional Attributes

You can specify the following optional attributes for an IP Monitoring job:

- [ip_port](#) (see page 437)
- [ip_status](#) (see page 438)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [monitor_mode](#) (see page 488)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor an IP Address for a Stopped Status

This example monitors a device with DNS name myhost. When the device stops running, the job completes.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: myhost
ip_status: STOPPED
monitor_mode: WAIT
```

JMS Publish Jobs

You can define a JMS Publish (JMSPUB) job to send a message to a queue or publish a message to a topic.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

JMS Publish jobs support the TextMessage and ObjectMessage message types.

Note: The StreamMessage, BytesMessage, and MapMessage message types are not supported.

Required Attributes

To define a JMS Publish job, you must specify the following attributes:

- [job_type: JMSPUB](#) (see page 464)
- [machine](#) (see page 473)
- [connection_factory](#) (see page 351)
- [destination_name](#) (see page 369)
- [initial_context_factory](#) (see page 429)
- [j2ee_parameter](#) (see page 444)
- [message_class](#) (see page 481)
- [provider_url](#) (see page 560)

Optional Attributes

You can specify the following optional attributes for a JMS Publish job:

- [j2ee_user](#) (see page 452)
- [job_class](#) (see page 458)
- [use_topic](#) (see page 737)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Publish a Message to a WebSphere MQ server

This example publishes the message "this is my message" to the queue named Queue. The Java class of the message is String. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere MQ server and 2809 is the ORB port. The job uses the cyberuser JNDI user name to gain access to the connection factory named ConnectionFactory.

```
insert_job: publish
job_type: JMSPUB
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
connection_factory: ConnectionFactory
destination_name: Queue
use_topic: FALSE
message_class: String
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="this is my message"
```

Note: The agent does not support JMS messaging on IBM WebSphere. If you have IBM WebSphere MQ, your agent administrator can set up the agent plug-in to run JMS Publish and JMS Subscribe for JMS queues. JMS topics are not supported on IBM WebSphere MQ.

JMS Subscribe Jobs

You can define a JMS Subscribe (JMSSUB) job to consume messages from a queue or topic.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

JMS Subscribe jobs support the TextMessage and ObjectMessage message types.

Note: The StreamMessage, BytesMessage, and MapMessage message types are not supported.

Required Attributes

To define a JMS Subscribe job, you must specify the following attributes:

- [job_type: JMSSUB](#) (see page 464)
- [machine](#) (see page 473)
- [connection_factory](#) (see page 351)
- [destination_name](#) (see page 369)
- [initial_context_factory](#) (see page 429)
- [provider_url](#) (see page 560)

Optional Attributes

You can specify the following optional attributes for a JMS Subscribe job:

- [continuous](#) (see page 353)
- [destination_file](#) (see page 367)
- [filter](#) (see page 383)
- [j2ee_user](#) (see page 452)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [use_topic](#) (see page 737)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor a Queue on a WebLogic Application Server

This example continuously monitors the queue named Queue (residing on WebLogic) for a message matching the filter criteria. The consumed messages from the queue are stored in the file /export/home/user1/outputfile1. The service provider's URL is t3://172.24.0.0:7001, where 172.24.0.0 is the IP address of the WebLogic Application server and 7001 is the ORB port.

```
insert_job: monitor
job_type: JMSSUB
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://172.24.0.0:7001"
connection_factory: ConnectionFactory
destination_name: Queue
continuous: Y
filter: abc\s...\\s[a-zA-Z]+\sFilter![\sa-zA-Z0-9]+
use_topic: FALSE
destination_file: /export/home/user1/outputfile1
j2ee_user: cyberuser
```

In this example, the regular expression used as the filter criteria can be defined as follows:

abc\s...\\s[a-zA-Z]+\sFilter![\sa-zA-Z0-9]+

abc\s

Specifies the text abc, followed by white space.

...\\s

Specifies any three characters, followed by white space.

[a-zA-Z]+\s

Specifies at least one letter, followed by white space.

Filter![\sa-zA-Z0-9]+

Specifies the text Filter!, followed by at least one of the following: white space or digit or lower case letter.

Example: abc vvv B Filter! 95

JMX-MBean Attribute Get Jobs

You can define a JMX-MBean Attribute Get (JMXMLAG) job to query a JMX server for the value of an MBean attribute. The returned value is stored on the computer where the agent resides. You can specify a success pattern to determine the job's success or failure. If the returned attribute value matches the success pattern, the job completes successfully; otherwise, it fails.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Attribute Get job, you must specify the following attributes:

- [job_type: JMXMLAG](#) (see page 464)
- [machine](#) (see page 473)
- [mbean_attr](#) (see page 477)
- [mbean_name](#) (see page 478)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Attribute Get job:

- [job_class](#) (see page 458)
- [jmx_user](#) (see page 457)
- [success_pattern](#) (see page 696)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Query a JMX Server for the Value of an MBean Attribute

Suppose that you want to know the value of the cachesize attribute for the Config MBean. The URL for the JMX server is service:jmx:rmi:///jndi/rmi://localhost:9999/server, where localhost is the host name and 9999 is the port number.

```
insert_job: query
job_type: JMXMAS
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_attr: cachesize
```

JMX-MBean Attribute Set Jobs

You can define a JMX-MBean Attribute Set (JMXXMAS) job to change the value of an MBean attribute on a JMX server. You can specify a set value for the attribute or use the serialized Java object passed by another job. When the attribute is set, the job returns the original attribute value as output. You can specify a success pattern to determine the job's success or failure. If the job's output matches the success pattern, the job completes successfully; otherwise, it fails.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Attribute Set job, you must specify the following attributes:

- [job_type: JMXXMAS](#) (see page 464)
- [machine](#) (see page 473)
- [mbean_attr](#) (see page 477)
- [mbean_name](#) (see page 478)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Attribute Set job:

- [jmxa_parameter](#) (see page 456)
- [jmxa_user](#) (see page 457)
- [job_class](#) (see page 458)
- [success_pattern](#) (see page 696)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Change the Value of an MBean Attribute

Suppose that you want to set the value of the State attribute of the cdc.jmx.SimpleDynamic MBean to the serialized Java object returned by a JMX-MBean Attribute Set job named size.

```
insert_job: change
job_type: JMXTMAS
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic"
mbean_attr: State
jmxt_parameter: payload_job=size
condition: success(size)
```

JMX-MBean Create Instance Jobs

You can define a JMX-MBean Create Instance (JMXMC) job to create an MBean on a JMX server.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Create Instance job, you must specify the following attributes:

- [job_type: JMXMC](#) (see page 464)
- [machine](#) (see page 473)
- [class_name](#) (see page 322)
- [mbean_name](#) (see page 478)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Create Instance job:

- [jmx_parameter](#) (see page 456)
- [jmx_user](#) (see page 457)
- [job_class](#) (see page 458)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Create an MBean Instance on a JMX Server

Suppose that you want to create an MBean instance on a JMX server. The job uses the cdc.jmx.SimpleDynamic class. The constructor of the class takes a single string parameter with the value "Hello".

```
insert_job: create
job_type: JMXML
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic"
class_name: cdc.jmx.SimpleDynamic
jmx_parameter: java.lang.String="Hello"
```

JMX-MBean Operation Jobs

You can define a JMX-MBean Operation (JMXMOP) job to invoke an operation on an MBean. You can specify one or more parameter values to pass to the operation. You can specify a success pattern to determine the job's success or failure. If the operation's output matches the success pattern, the job completes successfully; otherwise, it fails.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Operation job, you must specify the following attributes:

- [job_type: JMXMOP](#) (see page 464)
- [machine](#) (see page 473)
- [mbean_name](#) (see page 478)
- [mbean_operation](#) (see page 480)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Operation job:

- [jmx_parameter](#) (see page 456)
- [jmx_user](#) (see page 457)
- [job_class](#) (see page 458)
- [success_pattern](#) (see page 696)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Invoke an Operation on an MBean

Suppose that you want to invoke the resetmem operation on the config MBean to reset the value of the memory parameter to 50.

```
insert_job: reset
job_type: JMXMOP
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_operation: resetmem
jmx_parameter: Integer=50
```

Example: Pass Payload Producing Output as Input to Payload Consuming Job

Suppose that you want to use a JMX-MBean Operation job to invoke a method on an MBean and pass the output of the method as input to another JMX-MBean Operation job.

In this example, the first job, test_JMXMOP2a, is a payload producing job. It takes a single input parameter and invokes the reset method on the MBean. The output of this job is stored as a serialized Java object on the computer where the agent resides.

The second job, test_JMXMOP2b, is a payload consuming job. It takes two input parameters: the string "Hello" and the serialized Java object produced by the first job. The two input parameters are passed to the reset method, which is invoked on the MBean.

```
insert_job: test_JMXMOP2a
machine: localhost
job_type: JMXMOP
url: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:type=SimpleStandard,index=1"
mbean_operation: reset
jmx_parameter: String="Hello"

insert_job: test_JMXMOP2b
machine: localhost
job_type: JMXMOP
url: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:type=SimpleStandard,index=1"
mbean_operation: reset
jmx_parameter: String="Hello", payload_job=test_JMXMOP2a
condition: S(test_JMXMOP2a)
```

JMX-MBean Remove Instance Jobs

You can define a JMX-MBean Remove Instance (JM XMREM) job to remove an MBean from a JMX server.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Remove Instance job, you must specify the following attributes:

- [job_type: JM XMREM](#) (see page 464)
- [machine](#) (see page 473)
- [mbean_name](#) (see page 478)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Remove Instance job:

- [jm x user](#) (see page 457)
- [job_class](#) (see page 458)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Remove an MBean Instance from a JMX Server

Suppose that you want to remove an MBean instance.

```
insert_job: remove
job_type: JM XMREM
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index/CreateIns1,type=cdc.jmx.SimpleDynamic"
```

JMX-MBean Subscribe Jobs

You can define a JMX-MBean Subscribe (JMXSUB) job to monitor an MBean for a single notification or monitor continuously for notifications. You can filter the notifications the job monitors by attributes or by type of notifications.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a JMX-MBean Subscribe job, you must specify the following attributes:

- [job_type: JMXSUB](#) (see page 464)
- [machine](#) (see page 473)
- [mbean_name](#) (see page 478)
- [URL](#) (see page 736)

Optional Attributes

You can specify the following optional attributes for a JMX-MBean Subscribe job:

- [continuous](#) (see page 353)
- [filter](#) (see page 383)
- [filter_type](#) (see page 385)
- [jmx_user](#) (see page 457)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor for a Change to a Specific MBean Attribute

Suppose that you want to monitor for a change to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. When the cachesize attribute changes, the job completes.

```
insert_job: change
job_type: JMXSUB
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
filter_type: Attributes
filter: cachesize
```

Example: Monitor for Changes to Any MBean Attribute

Suppose that you want to set up continuous monitoring for changes to any attribute of the MBean named Config. Each time an attribute changes, an alert is written to the scheduler log file.

```
insert_job: change
job_type: JMXSUB
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
continuous: Y
filter_type: Types
filter: jmx.attribute.change
```

Oracle E-Business Suite Copy Single Request Jobs

You can define an Oracle E-Business Suite Copy Single Request (OACOPY) job to copy an existing single request defined on Oracle E-Business Suite and run it under the agent. When the job runs, it can override values in the original definition with values specified on the agent or in the job definition.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Oracle E-Business Suite.

Required Attributes

To define an Oracle E-Business Suite Copy Single Request job, you must specify the following attributes:

- [job_type: OACOPY](#) (see page 464)
- [machine](#) (see page 473)
- [request_id](#) (see page 599)

Important! All Oracle E-Business Suite jobs require a responsibility name. You must define this value; otherwise, the job fails. To define this value, you can specify a default responsibility name using the oa.default.responsibility parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_resp attribute in a job definition, which overrides the default responsibility name.

All Oracle E-Business Suite jobs require a user name. You must define this value; otherwise, the job fails. To define this value, you can specify a default user name using the oa.default.user parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_user attribute in a job definition, which overrides the default user name.

Optional Attributes

You can specify the following optional attributes for an Oracle E-Business Suite Copy Single Request job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [oracle_mon_children](#) (see page 519)
- [oracle_mon_children_delay](#) (see page 520)
- [oracle_resp](#) (see page 531)
- [oracle_user](#) (see page 536)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Copy a Single Request

Suppose that you want to copy an existing single request defined on Oracle E-Business Suite. In this example, the job copies the single request job with request ID 2255470 and overrides the Oracle E-Business Suite user name and responsibility name defined in the agentparm.txt file.

```
insert_job: oacopy_single
job_type: oacopy
machine: oaagent
request_id: 2255470
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

Oracle E-Business Suite Request Set Jobs

You can define an Oracle E-Business Suite Request Set (OASET) job to run multiple programs in an Oracle E-Business Suite application. To define an OASET job, you require the following information from the original Oracle E-Business Suite request set:

- Display name or short name of the Oracle E-Business Suite application the request set belongs to
- Request set name
- User name that the job runs under
- Responsibility name

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Oracle E-Business Suite.

Required Attributes

To define an Oracle E-Business Suite Request Set job, you must specify the following attributes:

- [job_type: OASET](#) (see page 464)
- [machine](#) (see page 473)
- [oracle_appl_name](#) (see page 513)
- [oracle_req_set](#) (see page 530)

Important! All Oracle E-Business Suite jobs require a responsibility name. You must define this value; otherwise, the job fails. To define this value, you can specify a default responsibility name using the oa.default.responsibility parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_resp attribute in a job definition, which overrides the default responsibility name.

All Oracle E-Business Suite jobs require a user name. You must define this value; otherwise, the job fails. To define this value, you can specify a default user name using the oa.default.user parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_user attribute in a job definition, which overrides the default user name.

Optional Attributes

You can specify the following optional attributes for an Oracle E-Business Suite Request Set job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [oracle_appl_name_type](#) (see page 515)
- [oracle_mon_children](#) (see page 519)
- [oracle_mon_children_delay](#) (see page 520)
- [oracle_print_copies](#) (see page 522)
- [oracle_print_style](#) (see page 523)
- [oracle_printer](#) (see page 525)
- [oracle_programdata](#) (see page 528)
- [oracle_resp](#) (see page 531)
- [oracle_save_output](#) (see page 533)
- [oracle_use_arg_def](#) (see page 534)
- [oracle_user](#) (see page 536)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a Request Set

This example runs a request set named FNDRSSUB1310 on the oaagent agent. The request set belongs to the application with the short name BIS in Oracle E-Business Suite. The job runs under the SYSADMIN user with Business Intelligence Super User, Progress S&L responsibility. The job's output is saved.

```
insert_job: oaset_resp
job_type: oaset
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: BIS
oracle_req_set: FNDRSSUB1310
oracle_user: SYSADMIN
oracle_resp: "Business Intelligence Super User, Progress S&L"
oracle_save_output: Y
```

Oracle E-Business Suite Single Request Jobs

You can define an Oracle E-Business Suite Single Request (OASG) job to run a single program in an Oracle E-Business Suite application. To define an OASG job, you require the following information from the original Oracle E-Business Suite single request:

- Display name or short name of the Oracle E-Business Suite application the single request belongs to
- Program short name
- User ID the job runs under
- Responsibility name

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Oracle E-Business Suite.

Required Attributes

To define an Oracle E-Business Suite Single Request job, you must specify the following attributes:

- [job_type: OASG](#) (see page 464)
- [machine](#) (see page 473)
- [oracle_appl_name](#) (see page 513)
- [oracle_program](#) (see page 527)

Important! All Oracle E-Business Suite jobs require a responsibility name. You must define this value; otherwise, the job fails. To define this value, you can specify a default responsibility name using the oa.default.responsibility parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_resp attribute in a job definition, which overrides the default responsibility name.

All Oracle E-Business Suite jobs require a user name. You must define this value; otherwise, the job fails. To define this value, you can specify a default user name using the oa.default.user parameter in the agent's agentparm.txt file. Alternatively, you can specify the oracle_user attribute in a job definition, which overrides the default user name.

Optional Attributes

You can specify the following optional attributes for an Oracle E-Business Suite Single Request job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [oracle_appl_name_type](#) (see page 515)
- [oracle_args](#) (see page 516)
- [oracle_desc](#) (see page 518)
- [oracle_mon_children](#) (see page 519)
- [oracle_mon_children_delay](#) (see page 520)
- [oracle_print_copies](#) (see page 522)
- [oracle_print_style](#) (see page 523)
- [oracle_printer](#) (see page 525)
- [oracle_resp](#) (see page 531)
- [oracle_save_output](#) (see page 533)
- [oracle_use_arg_def](#) (see page 534)
- [oracle_user](#) (see page 536)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a Single Request Program

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The history of the program is included in the description.

```
insert_job: oasg_disp
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_desc: CONFIRMED REQ 2010808 by TYB212
```

PeopleSoft Jobs

You can define a PeopleSoft (PS) job to schedule workload to run in PeopleSoft. The job runs a PeopleSoft request or a collection of requests. In the job definition, you can set the output type and format of a report. For email and web output types, you can set various distribution properties such as the recipients and message text. You can add parameters to the run control table that contains the PeopleSoft run parameters.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for PeopleSoft.

Required Attributes

To define a PeopleSoft job, you must specify the following attributes:

- [job_type: PS](#) (see page 464)
- [machine](#) (see page 473)
- [ps_process_name](#) (see page 585)
- [ps_process_type](#) (see page 586)

Optional Attributes

You can specify the following optional attributes for a PeopleSoft job:

- [envvars](#) (see page 378)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [ps_args](#) (see page 565)
- [ps_dest_format](#) (see page 566)
- [ps_dest_type](#) (see page 568)
- [ps_detail_folder](#) (see page 570)
- [ps_dlist_roles](#) (see page 571)
- [ps_dlist_users](#) (see page 572)
- [ps_email_address](#) (see page 575)
- [ps_email_address_expanded](#) (see page 576)
- [ps_email_log](#) (see page 577)
- [ps_email_subject](#) (see page 578)
- [ps_email_text](#) (see page 579)
- [ps_email_web_report](#) (see page 580)
- [ps_operator_id](#) (see page 581)
- [ps_output_dest](#) (see page 582)
- [ps_restarts](#) (see page 587)
- [ps_run_cntrl_args](#) (see page 588)
- [ps_run_cntrl_id](#) (see page 591)
- [ps_run_control_table](#) (see page 592)
- [ps_server_name](#) (see page 594)
- [ps_skip_parm_updates](#) (see page 595)
- [ps_time_zone](#) (see page 597)

Notes:

- All PeopleSoft jobs require a run control ID. You must define this value; otherwise, the job fails. Your agent administrator can specify a default run control ID using the ps.default.runCtlId parameter on the agent, or you can specify the ps_run_cntrl_id attribute in the job definition.
- All PeopleSoft jobs require a valid operator ID. The operator ID and corresponding password can be defined on the agent using the ps.default.oprId and ps.default.oprPassword parameters in the agentparm.txt file. You can specify the operator ID in a job definition using the ps_operator_id attribute.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a PeopleSoft Request

This example runs an SQR Report. The report is formatted as PDF and outputted to a file.

```
insert_job: ps_file
job_type: ps
machine: psagent
ps_process_name: XRFWIN
ps_process_type: SQR Report
ps_dest_type: FILE
ps_dest_format: PDF
ps_output_dest: /export/home/PSoutput/report1.pdf
ps_run_cntrl_id: test
owner: VP1@ps1
```

POJO Jobs

You can define a POJO job to create a Java object instance with no arguments, invoke a method on the object instance, and store the method's output.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and either CA WA Agent for Application Services or CA WA Agent for Web Services.

Required Attributes

To define a POJO job, you must specify the following attributes:

- [job_type: POJO](#) (see page 464)
- [machine](#) (see page 473)
- [class_name](#) (see page 322)
- [method_name](#) (see page 482)

Optional Attributes

You can specify the following optional attributes for a POJO job:

- [destination_file](#) (see page 367)
- [j2ee_parameter](#) (see page 444)
- [job_class](#) (see page 458)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Invoke a Method on a Java Object Instance

Suppose that you want to define a POJO job that creates a Java String with value "5" and calls the parseInt method with the created Java String object as an argument. The parseInt method returns a Java Integer object.

```
insert_job: ignore
job_type: POJO
machine: appagent
class_name: java.lang.Integer
method_name: parseInt
j2ee_parameter: java.lang.String=5
```

Process Monitoring Jobs

You can define a Process Monitoring (OMP) job to monitor the status of a process on the computer where the agent is installed.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a Process Monitoring job, you must specify the following attributes:

- [job_type: OMP](#) (see page 464)
- [machine](#) (see page 473)
- [process_name](#) (see page 552)

Optional Attribute

You can specify the following optional attribute for a Process Monitoring job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [monitor_mode](#) (see page 488)
- [process_status](#) (see page 555)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor a Running Process

This example monitors the process with ID 2568. If the process is running, the job completes successfully. If the process is not running, the job continues monitoring it until the process starts running.

```
insert_job: omp_unix
job_type: OMP
machine: unixagt
process_name: 2568
process_status: RUNNING
monitor_mode: WAIT
```

RMI Jobs

You can define an RMI (JAVARMI) job to call a method on a remote server and store the method's output.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define an RMI job, you must specify the following attributes:

- [job_type: JAVARMI](#) (see page 464)
- [machine](#) (see page 473)
- [method_name](#) (see page 482)
- [remote_name](#) (see page 598)

Optional Attributes

You can specify the following optional attributes for an RMI job:

- [destination_file](#) (see page 367)
- [j2ee_parameter](#) (see page 444)
- [job_class](#) (see page 458)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define a Job to Start a Remote Server Immediately

Suppose that you want to invoke a method that starts a remote server using remote object activation. You want the server to start immediately.

```
insert_job: start
job_type: JAVARMI
machine: appagent
remote_name: "rmi://remotehost/Test"
method_name: startserver
j2ee_parameter: String="now"
```

SAP Batch Input Session Jobs

You can define an SAP Batch Input Session (SAPBDC) job to import large amounts of data from external systems to the SAP system. You create Batch Input Session jobs on the SAP system.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP Batch Input Session job, you must specify the following attributes:

- [job_type: SAPBDC](#) (see page 464)
- [machine](#) (see page 473)
- [sap_job_name](#) (see page 620)
- [sap_step_parms](#) (see page 638)

Optional Attributes

You can specify the following optional attributes for an SAP Batch Input Session job:

- [bdc_err_rate](#) (see page 307)
- [bdc_ext_log](#) (see page 308)
- [bdc_proc_rate](#) (see page 309)
- [bdc_system](#) (see page 310)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_job_class](#) (see page 618)
- [sap_lang](#) (see page 622)
- [sap_office](#) (see page 624)
- [sap_recipients](#) (see page 633)
- [sap_release_option](#) (see page 635)
- [sap_rfc_dest](#) (see page 636)
- [sap_target_sys](#) (see page 649)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP Batch Input Session Job

This example runs the ZBDCTEST ABAP that creates a Batch Input Session (BDC ABAP) on the default SAP system defined to the agent. The job runs as soon as possible after the job is defined. After this job runs, the BDC job starts the data transfer.

```
insert_job: bdcjob
job_type: SAPBDC
machine: sapagent
owner: WAAESAP@sapagent
sap_job_name: ZBDCTEST
sap_release_option: A
sap_step_parms: abap_name="ZBDCTEST"
```

SAP BW InfoPackage Jobs

You can define an SAP BW InfoPackage (SAPBWIP) job to transfer data from any data source into an SAP Business Warehouse system. When the job runs, the data is transferred.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP BW InfoPackage job, you must specify the following attributes:

- [job_type: SAPBWIP](#) (see page 464)
- [machine](#) (see page 473)
- [sap_info_pack](#) (see page 616)

Optional Attributes

You can specify the following optional attributes for an SAP BW InfoPackage job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_ext_table](#) (see page 614)
- [sap_job_name](#) (see page 620)
- [sap_lang](#) (see page 622)
- [sap_rfc_dest](#) (see page 636)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP BW InfoPackage Job

This example runs the Business Warehouse InfoPackage OPAK_D2XZMZ1HD5WFVFL3EN1NVIT4V at the SAP destination, SM1.

```
insert_job: InfoBW2
job_type: SAPBWIIP
machine: sapagent
owner: WAAESAP@sapagent
sap_job_name: InfoBW2
sap_rfc_dest: SM1
sap_info_pack: OPAK_D2XZMZ1HD5WFVFL3EN1NVIT4V
```

SAP BW Process Chain Jobs

You can define an SAP BW Process Chain (SAPBWPC) job to run a sequence of background processes on the SAP system. Some SAP processes trigger events that can start other processes. A job runs the individual processes in the chain as job steps.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP BW Process Chain job, you must specify the following attributes:

- [job_type: SAPBWPC](#) (see page 464)
- [machine](#) (see page 473)
- [sap_chain_id](#) (see page 610)

Optional Attributes

You can specify the following optional attributes for an SAP BW Process Chain job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_lang](#) (see page 622)
- [sap_rfc_dest](#) (see page 636)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP BW Process Chain Job

This example defines the SAPBWPC3 job that runs the DEMO_CHAIN1 Process Chain on the SAP system. The language used to log on to the SAP system is English.

```
insert_job: SAPBWPC3
job_type: SAPBWPC
machine: sapagent
owner: WAAESAP@sapagent
sap_chain_id: DEMO_CHAIN1
sap_lang: EN
sap_rfc_dest: SM1
```

SAP Data Archiving Jobs

You can define an SAP Data Archiving (SAPDA) job to store information described in an SAP Archiving Object into an SAP data archive.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP Data Archiving job, you must specify the following attributes:

- [job_type: SAPDA](#) (see page 464)
- [machine](#) (see page 473)
- [arc_obj_name](#) (see page 297)
- [arc_obj_variant](#) (see page 298)

Optional Attributes

You can specify the following optional attributes for an SAP Data Archiving job:

- [arc_parms](#) (see page 299)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_lang](#) (see page 622)
- [sap_print_parms](#) (see page 625)
- [sap_rfc_dest](#) (see page 636)
- [sap_target_sys](#) (see page 649)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP Data Archiving Job

This example defines a job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
insert_job: SAPDA_job
job_type: SAPDA
machine: sapagent
owner: WAAESAP@sapagent
arc_obj_name: BC_ARCHIVE
arc_obj_variant: BC_ARCVARIANT
sap_print_parms: dest=LP01,prt_arc_mode=PRINT
```

SAP Event Monitor Jobs

You can define an SAP Event Monitor (SAPEVT) job to schedule workload based on the activity of an SAP event or trigger an SAP event at the appropriate time in your schedule.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP Event Monitor job, you must specify the following attributes:

- [job_type: SAPEVT](#) (see page 464)
- [machine](#) (see page 473)
- [sap_event_id](#) (see page 612)

Optional Attributes

You can specify the following optional attributes for an SAP Event Monitor job:

- [continuous](#) (see page 353)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_event_parm](#) (see page 613)
- [sap_is_trigger](#) (see page 617)
- [sap_lang](#) (see page 622)
- [sap_rfc_dest](#) (see page 636)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Trigger an SAP Event

This example defines an SAPEVT job. The job triggers the SAP_TEST event and the SAP job dependencies waiting on the SAP_TEST event are satisfied.

```
insert_job: SAPEVT_job
job_type: SAPEVT
machine: sapagent
owner: WAAESAP@sapagent
sap_event_id: SAP_TEST
sap_rfc_dest: BI1
sap_is_trigger: Y
```

SAP Job Copy Jobs

You can define an SAP Job Copy (SAPJC) job to schedule an existing SAP R/3 job.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP. This job type requires the SAP XBP 2.0 interface or higher to be available.

Required Attributes

To define an SAP Job Copy job, you must specify the following attributes:

- [job_type: SAPJC](#) (see page 464)
- [machine](#) (see page 473)
- [sap_job_count](#) (see page 619)
- [sap_job_name](#) (see page 620)

Optional Attributes

You can specify the following optional attributes for an SAP Job Copy job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_fail_msg](#) (see page 615)
- [sap_lang](#) (see page 622)
- [sap_mon_child](#) (see page 623)
- [sap_release_option](#) (see page 635)
- [sap_rfc_dest](#) (see page 636)
- [sap_step_num](#) (see page 637)
- [sap_success_msg](#) (see page 648)
- [sap_target_jobname](#) (see page 648)
- [sap_target_sys](#) (see page 649)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP Job Copy Job

This example defines an SAP Job Copy job that copies the AM job with job count 11331500 and runs the new copy.

```
insert_job: SAPJC_job
job_type: SAPJC
owner: WAAESAP@sapagent
machine: sapagent
sap_job_name: AM
sap_job_count: 11331500
```

SAP Process Monitor Jobs

You can define an SAP Process Monitor (SAPPM) job to monitor for a specific SAP process status and end after detecting a process. You can also use SAP Process Monitor jobs to set up predecessor or dependent job relationships with other jobs or SAP processes.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP Process Monitor job, you must specify the following attributes:

- [job_type: SAPPM](#) (see page 464)
- [machine](#) (see page 473)
- [sap_process_status](#) (see page 630)

Optional Attributes

You can specify the following optional attributes for an SAP Process Monitor job:

- [continuous](#) (see page 353)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_abap_name](#) (see page 609)
- [sap_client](#) (see page 611)
- [sap_lang](#) (see page 622)
- [sap_proc_type](#) (see page 631)
- [sap_proc_user](#) (see page 632)
- [sap_process_client](#) (see page 629)
- [sap_rfc_dest](#) (see page 636)
- [sap_target_sys](#) (see page 649)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP Process Monitor Job

This example defines an SAP Process Monitor job that monitors for the ABAP program ZMYABAP to change to the RUNNING status.

```
insert_job: test_SAPP  
job_type: SAPP  
machine: sapagent  
sap_abap_name: ZMYABAP  
sap_process_status: RUNNING  
sap_rfc_dest: BI1  
owner: WAAESAP@sapagent
```

SAP R/3 Jobs

You can define an SAP R/3 job to schedule an SAP R/3 job on your SAP system.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Attributes

To define an SAP R/3 job, you must specify the following attributes:

- [job_type: SAP](#) (see page 464)
- [machine](#) (see page 473)
- [sap_job_name](#) (see page 620)
- [sap_step_parms](#) (see page 638)

Optional Attributes

You can specify the following optional attributes for an SAP R/3 job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [sap_client](#) (see page 611)
- [sap_fail_msg](#) (see page 615)
- [sap_job_class](#) (see page 618)
- [sap_lang](#) (see page 622)
- [sap_mon_child](#) (see page 623)
- [sap_office](#) (see page 624)
- [sap_recipients](#) (see page 633)
- [sap_release_option](#) (see page 635)
- [sap_rfc_dest](#) (see page 636)
- [sap_success_msg](#) (see page 647)
- [sap_target_sys](#) (see page 649)

Notes:

- We recommend that you limit the number of steps (ABAPs) to one per job. If you run a job and one of the ABAPs fails, the job is marked as failed. If the ABAP fails, you cannot re-run the ABAP without re-running the entire job.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define an SAP R/3 Job With One Step

This example runs the SAP R/3 job named BI_WRITE_PROT_TO_APPLLOG. The job runs the SAP program (ABAP) named RSBATCH_WRITE_PROT_TO_APPLLOG. The owner is defined in CA Workload Automation AE using the autosys_secure command.

```
insert_job: test_SAP
job_type: SAP
machine: sapagent
sap_job_name: BI_WRITE_PROT_TO_APPLLOG
sap_rfc_dest: BI1
sap_step_parms: abap_lang=EN,abap_name=RSBATCH_WRITE_PROT_TO_APPLLOG
owner: WAAESAP@sapagent
```

Secure Copy Jobs

You can define a Secure Copy (SCP) job to transfer binary files using the Secure Copy Protocol.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a Secure Copy job, you must specify the following attributes:

- [job_type: SCP](#) (see page 464)
- [machine](#) (see page 473)
- [scp_local_name](#) (see page 650)
- [scp_remote_dir](#) (see page 654)
- [scp_remote_name](#) (see page 656)
- [scp_server_name](#) (see page 658)

Optional Attributes

You can specify the following optional attributes for a Secure Copy job:

- [job_class](#) (see page 458)
- [scp_local_user](#) (see page 652)
- [scp_protocol](#) (see page 653)
- [scp_server_port](#) (see page 659)
- [scp_target_os](#) (see page 660)
- [scp_transfer_direction](#) (see page 661)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Upload a File Using the Secure File Transfer Protocol

This example uploads the logs.tar file to the /u/tmp directory on the hpsupport server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_upload
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: hpsupport
scp_remote_dir: /u/tmp
scp_remote_name: logs.tar
scp_local_name: "D:\temp\logs.tar"
scp_protocol: SFTP
owner: causer@WINAGENT
```

Example: Upload Multiple Files Using the Secure File Transfer Protocol

This example uploads the files in the c:\temp\upload directory to the /u1/build/uploaded directory on the aixunix server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_up_mult
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: aixunix
scp_remote_dir: /u1/build/uploaded
scp_remote_name: "*"
scp_local_name: "c:\temp\upload\*"
scp_protocol: SFTP
owner: causer@WINAGENT
```

Session Bean Jobs

You can define a Session Bean (SESSBEAN) job to access a stateless or stateful session bean, invoke a method on the bean, and return the results.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Required Attributes

To define a Session Bean job, you must specify the following attributes:

- [job_type: SESSBEAN](#) (see page 464)
- [machine](#) (see page 473)
- [bean_name](#) (see page 311)
- [initial_context_factory](#) (see page 429)
- [method_name](#) (see page 482)
- [provider_url](#) (see page 560)

In addition, to access stateful session beans, you require the following attributes:

- [create_method](#) (see page 357)
- [create_parameter](#) (see page 359)

Optional Attributes

You can specify the following optional attributes for a Session Bean job:

- [destination_file](#) (see page 367)
- [j2ee_parameter](#) (see page 444)
- [j2ee_user](#) (see page 452)
- [job_class](#) (see page 458)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Invoke a Method on a Stateless Session Bean

Suppose that you want to invoke the reverse method on the CybEJBTTestBean stateless session bean. The reverse method has one parameter, with type java.lang.String and value a23. The output from the reverse method is saved in the C:\Makapt15 file. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere application server and 2809 is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

Example: Invoke a Method on a Stateful Session Bean

Suppose that you want to access a stateful session bean for an online shopping cart. The createaddbook method creates the ShoppingCart stateful bean for the duration of the job. The addbook method adds books to the shopping cart using the book's ISBN number. In this example, the Session Bean job adds two books to the shopping cart with ISBN numbers 1551929120 and 1582701709. When the job runs, two books are added to the shopping cart.

```
insert_job: addbook
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: ShoppingCart
create_method: createaddbook
method_name: addbook
create_parameter: String="ISBN"
j2ee_parameter: Integer[2]="1551929120,1582701709"
```

SQL Jobs

You can define an SQL job to run an SQL query against an Oracle, SQL Server, Sybase, or DB2 database. When the job runs, the SQL statement is invoked and the results are stored in an output file or job spool file. You can also add criteria to the job definition to test the query result. If the result matches the criteria, the job completes successfully. Otherwise, the job fails.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Attributes

To define an SQL job, you must specify the following attributes:

- [job_type: SQL](#) (see page 464)
- [machine](#) (see page 473)
- [sql_command](#) (see page 674)

Optional Attributes

You can specify the following optional attributes for an SQL job:

- [connect_string](#) (see page 349)
- [destination_file](#) (see page 367)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [success_criteria](#) (see page 693)
- [user_role](#) (see page 738)

Notes:

- All database jobs require a database resource location. You must define this value; otherwise, the job fails. Your agent administrator can specify a default location using the db.default.url parameter on the agent, or you can specify the connect_string attribute in the job definition.
- All database jobs require a user ID that has the appropriate permissions to access the information in the database. The job runs under that user ID. The [owner_attribute](#) (see page 538) in the job definition specifies the user ID (the default is the user who invokes jil to define the job).
- Windows authentication is not supported.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Delete a Row from an Oracle Database Table

This example deletes a row from the emp table for the employee with ename robert.

```
insert_job: deletejob
job_type: SQL
machine: DB_agent
sql_command: DELETE FROM EMP WHERE ENAME=' robert'
owner: scott@orcl
connect_string:"jdbc:oracle:thin:@myhost:1521:orcl"
```

Text File Reading and Monitoring Jobs

You can define a Text File Reading and Monitoring (OMTF) job to search a text file on a Windows, UNIX, or i5/OS computer for a text string. For example, you can monitor a log file for an error message after a script executes.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Attributes

To define a Text File Reading and Monitoring job, you must specify the following attributes:

- [job_type: OMTF](#) (see page 464)
- [machine](#) (see page 473)
- [text_file_filter](#) (see page 707)
- [text_file_name](#) (see page 715)

Optional Attributes

You can specify the following optional attributes for a Text File Reading and Monitoring job:

- [encoding](#) (see page 375)
- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [lower_boundary](#) (see page 467)
- [monitor_mode](#) (see page 488)
- [text_file_filter_exists](#) (see page 710)
- [text_file_mode](#) (see page 711)
- [time_format](#) (see page 717)
- [time_position](#) (see page 720)
- [upper_boundary](#) (see page 731)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor a File for a Text String

This example searches for a text string in the log text file on a Windows computer. The path to the text file is enclosed in quotation marks because the path contains a colon. The job searches for the string "ERROR MESSAGE" between lines 1234 and 1876. The job completes successfully if the string is found.

```
insert_job: textfile_job1
job_type: OMTF
machine: monagt
text_file_name: "c:\program files\agent\log"
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 1234
upper_boundary: 1876
monitor_mode: NOW
```

Web Service Jobs

You can define a Web Service (WBSVC) job to call an operation within a web service.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Web Services.

Required Attributes

To define a Web Service job, you must specify the following attributes:

- [job_type: WBSVC](#) (see page 464)
- [machine](#) (see page 473)
- [target_namespace](#) (see page 705)
- [wsdl_operation](#) (see page 788)

Optional Attributes

You can specify the following optional attributes for a Web Service job:

- [destination_file](#) (see page 367)
- [endpoint_URL](#) (see page 376)
- [job_class](#) (see page 458)
- [one_way](#) (see page 510)
- [port_name](#) (see page 549)
- [return_class_name](#) (see page 601)
- [return_namespace](#) (see page 602)
- [return_xml_name](#) (see page 604)
- [service_name](#) (see page 666)
- [success_pattern](#) (see page 696)
- [web_parameter](#) (see page 773)
- [web_user](#) (see page 776)
- [WSDL_URL](#) (see page 790)

Notes:

- In a Web Service job, if you specify the WSDL_URL attribute but not the endpoint_URL attribute, you must specify both the service_name and port_name attributes. For the job to run successfully without the endpoint_URL attribute, the agent must be running on the same computer as the application server such as WebLogic or JBoss. If you specify both the WSDL_URL and endpoint_URL attributes, then the service_name and port_name attributes are optional.
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Get a Company Stock Quote

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webservicex.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a java.lang.String object, which maps to the XML type string in the return namespace <http://www.webservicex.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

Example: Validate an Email Address in a Web Service Job

Suppose that you want to invoke a web service that validates an email address. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.net/ValidateEmail.asmx?wsdl>. The job calls the `IsValidEmail` operation within the `ValidateEmail` web service. When the job invokes the web service, the email address is passed to the operation. If the email address is valid, the operation returns true and the job completes successfully. If the email address is invalid, the operation returns false and the job fails.

```
insert_job: subscribe
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: ValidateEmail
port_name: ValidateEmailSoap
wsdl_operation: IsValidEmail
WSDL_URL: "http://www.webservicex.net/ValidateEmail.asmx?wsdl"
endpoint_URL: "http://www.webservicex.net/ValidateEmail.asmx"
web_parameter: xsd\string="john.smith@example.com"
return_class_name: java.lang.Boolean
return_xml_name: boolean
return_namespace: "http://www.webservicex.net"
success_pattern: true
```

Windows Event Log Monitoring Jobs

You can define a Windows Event Log Monitoring (OMEL) job to monitor a Windows event log on a local computer. The monitor returns the most recent event available or continuously monitors for events in a particular Windows event log.

The Windows Event Log Monitoring job only monitors event logs maintained by the operating system and available in the Event Viewer. For more information about the Windows event properties stored in Windows event logs, view the logs in the Event Viewer on your local machine.

Note: To run these jobs, your system requires CA WA Agent for Windows.

Required Attributes

To define a Windows Event Log Monitoring job, you must specify the following attributes:

- [job_type: OMEL](#) (see page 464)
- [machine](#) (see page 473)
- [win_log_name](#) (see page 785)

Optional Attributes

You can specify the following optional attributes for a Windows Event Log Monitoring job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [monitor_mode](#) (see page 488)
- [win_event_category](#) (see page 777)
- [win_event_computer](#) (see page 778)
- [win_event_datetime](#) (see page 779)
- [win_event_description](#) (see page 780)
- [win_event_id](#) (see page 781)
- [win_event_op](#) (see page 782)
- [win_event_source](#) (see page 783)
- [win_event_type](#) (see page 784)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Monitor an Application Log That Occurs On or After a Specified Date

This example monitors an application log that occurs any time on or after January 12, 2010, 6:30 a.m. When the job finds an application log that occurs any time on or after that date and time, the job completes successfully.

```
insert_job: win_eventlog
job_type: OMEL
machine: winagent
win_log_name: Application
win_event_type: info
win_event_category: None
win_event_source: LLDSAPNT223
win_event_datetime: "20100112 06:30:00"
```

Windows Service Monitoring Jobs

You can define a Windows Service Monitoring (OMS) job to monitor services on a local Windows computer.

Note: To run these jobs, your system requires CA WA Agent for Windows.

Required Attributes

To define a Windows Service Monitoring job, you must specify the following attributes:

- [job_type: OMS](#) (see page 464)
- [machine](#) (see page 473)
- [win_service_name](#) (see page 786)

Optional Attributes

You can specify the following optional attribute for a Windows Service Monitoring job:

- [job_class](#) (see page 458)
- [job_terminator](#) (see page 461)
- [monitor_mode](#) (see page 488)
- [win_service_status](#) (see page 787)

Notes:

- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Check a Service Status Immediately

This example monitors the schedmanager service for a status of RUNNING. The job checks the status immediately and completes successfully if the service is running. If the service is not running, the job fails.

```
insert_job: oms_job3
job_type: OMS
machine: winagt
win_service_name: schedmanager
win_service_status: RUNNING
monitor_mode: NOW
```

z/OS Data Set Trigger Jobs

You can define a z/OS Data Set Trigger (ZOSDST) job to create dependencies on data set activities.

Note: To run these jobs, your system requires CA WA Agent for z/OS.

Required Attributes

To define z/OS Data Set Trigger job, you must specify the following attributes:

- [job_type: ZOSDST](#) (see page 464)
- [machine](#) (see page 473)
- [zos_dataset](#) (see page 792)

Optional Attributes

You can specify the following optional attributes for z/OS Data Set Trigger job:

- [zos_dsn_renamed](#) (see page 793)
- [zos_dsn_updated](#) (see page 794)
- [zos_explicit_dsn](#) (see page 795)
- [zos_ftp_direction](#) (see page 796)
- [zos_ftp_host](#) (see page 797)
- [zos_ftp_userid](#) (see page 798)
- [zos_trigger_by](#) (see page 800)
- [zos_trigger_on](#) (see page 802)
- [zos_trigger_type](#) (see page 803)

Notes:

- All z/OS jobs require a z/OS user ID. This user owns the job on z/OS. The owner attribute in the job definition specifies this user ID (the default is the user who invokes jil to define the job).
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Run a Compress Job After 100 Closures of a Data Set

Suppose that you want the z/OS Data Set Trigger job PROD.PAY_DATA to release a compress job named COPYJCL.COMPRESS after every 100 closures of the data set CYBER.COPY.JCL. The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: CYBER.COPY.JCL
zos_trigger_on: 100
```

Note: Define the compress job, COPYJCL.COMPRESS, as a successor to the z/OS Data Set Trigger job.

z/OS Manual Jobs

You can define a z/OS Manual (ZOSM) job to create dependencies on z/OS jobs that are submitted outside the scheduling manager, such as a job that is submitted manually by a user.

Note: To run these jobs, your system requires CA WA Agent for z/OS.

Required Attributes

To define z/OS Manual job, you must specify the following attributes:

- [job_type: ZOSM](#) (see page 464)
- [machine](#) (see page 473)
- [zos_jobname](#) (see page 799)

Optional Attributes

You can specify the following optional attributes for z/OS Manual job:

- [auth_string](#) (see page 303)
- [job_terminator](#) (see page 461)
- [search_bw](#) (see page 662)

Notes:

- All z/OS jobs require a z/OS user ID. This user owns the job on z/OS. The owner attribute in the job definition specifies this user ID (the default is the user who invokes jil to define the job).
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Post a z/OS Manual Job as Complete Based on the User ID

This example posts a z/OS Manual job as complete when the manually-submitted job ABC runs under user CYBER. The ZOS1 agent monitors job ABC.

```
insert_job: ABC_job
job_type: ZOSM
machine: ZOS1
zos_jobname: ABC
owner: zosuser
auth_string: CYBER
```

z/OS Regular Jobs

You can define a z/OS Regular (ZOS) job to schedule a z/OS job.

Note: To run these jobs, your system requires CA WA Agent for z/OS.

Required Attributes

To define z/OS Regular job, you must specify the following attributes:

- [job_type: ZOS](#) (see page 464)
- [machine](#) (see page 473)
- [jcl_library](#) (see page 454)

Optional Attributes

You can specify the following optional attributes for z/OS Regular job:

- [condition_code](#) (see page 346)
- [copy_jcl](#) (see page 356)
- [envvars](#) (see page 378)
- [jcl_member](#) (see page 455)

Note: If you specify a partitioned data set (PDS) in the jcl_library attribute, the jcl_member attribute is required.

- [job_terminator](#) (see page 461)

Notes:

- All z/OS jobs require a z/OS user ID. This user owns the job on z/OS. The owner attribute in the job definition specifies this user ID (the default is the user who invokes jil to define the job).
- You can also specify [optional common attributes](#) (see page 293) that apply to all job types.
- Default values for some attributes can be defined on the agent in the agentparm.txt file. If you specify the attribute in the job definition, it overrides the default value defined on the agent. For more information about possible default values, see the syntax and notes for the attributes you are using.

Example: Define a z/OS Regular Job

This example submits the JCL in member CYBDL01A in the CYBDL01.JCLLIB library.

```
insert_job: zos_job
job_type: ZOS
machine: zosagt
jcl_library: CYBDL01.JCLLIB
jcl_member: CYBDL01A
```

Chapter 4: JIL Job Definitions

This chapter describes the JIL subcommands and attributes that you can use to define and manage jobs. To define or manage a job, you specify the appropriate JIL subcommand and attribute statements to the `jil` command. JIL attributes define a job's properties. For example, the `job_type` attribute defines the type of job that you are creating or updating, and the `machine` attribute defines the name of a machine definition that describes where the job runs.

Note: When issuing commands or defining jobs that run on a different operating system (for example, Windows to UNIX or UNIX to Windows), you must use the syntax appropriate to the operating system of the agent machine where the job runs.

delete_box Subcommand—Delete a Box Job from the Database

The `delete_box` subcommand deletes the specified box job and all the jobs in that box from the database. If the box job or any jobs in it are already scheduled, they are deleted without running.

Note: To delete a box job but not the jobs in it, use the `delete_job` subcommand instead.

Syntax

This subcommand has the following format:

`delete_box: box_name`

box_name

Defines the name of a box job that is currently defined in the database.

Example: Delete a Box Job from the Database

This example deletes the box job `Box1` and all jobs in it:

`delete_box: Box1`

delete_job Subcommand—Delete a Job from the Database

The delete_job subcommand deletes the specified job from the database. If the job is already scheduled to run, it will not run.

If running jil in job verification mode, the delete_job subcommand checks the ujo_job_cond table and notifies you if the deleted job had dependent conditions. If the specified job is a box job, the box job is deleted and the jobs in the box become stand-alone jobs.

Note: To delete a box job and the jobs in it, use the delete_box subcommand instead.

Syntax

This subcommand has the following format:

`delete_job: job_name`

job_name

Defines the name of a job or box job that is currently defined in the database.

Limits: Up to 64 characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (_), hyphen (-), and pound (#); do not include embedded spaces or tabs

Example: Delete a Job from the Database

This example deletes the job Job1:

```
delete_job: Job1
```

insert_job Subcommand—Add a Job to the Database

The insert_job subcommand adds a job definition to the database. Depending on the type of job you are adding, the insert_job subcommand requires one or more additional attributes.

Note: You must define the machine using the insert_machine command before you can insert a job that uses that machine. The job will not insert if the machine definition does not exist.

Syntax

This subcommand has the following format:

```
insert_job: job_name  
job_name
```

Defines the name of the job that you want to schedule.

Limits: Up to 64 characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (_), hyphen (-), and pound (#); do not include embedded spaces or tabs

Example: Add a Command Job to the Database

This example creates the command job time_stamp to run on the real machine prod that runs the time_stamp.bat batch file:

```
insert_job: time_stamp  
machine: prod  
command: time_stamp.bat
```

The job_type attribute is optional when defining a command job. To specify a command job, enter:

```
job_type: CMD
```

Example: Add a File Watcher Job to the Database

This example creates the file watcher job EOD_batch_watch to run on the real machine prod that watches for a file named C:\myapps\EOD_batch.bat:

```
insert_job: EOD_batch_watch  
job_type: fw  
machine: prod  
watch_file: "C:\myapps\EOD_batch.bat"
```

Example: Add a Box Job to the Database

This example creates the box job end_of_day:

```
insert_job: end_of_day  
job_type: box
```

override_job Subcommand—Define a Job Override

The override_job subcommand defines a one-time override of the specified attributes. The override applies to the next run of the job you are defining.

If CA Workload Automation AE generates a RESTART event because of system problems, the job override is reissued until the job runs once or until the maximum number of retries set in the n_retrys attribute is reached. The override is then discarded.

The jil command does not accept overrides that result in an invalid job definition. For example, if a job definition has one starting condition, start_times, you cannot set the start_times attribute to NULL because removing the start condition makes the job definition invalid. In this situation, no start time can be calculated.

CA Workload Automation AE assigns an over_num value to the override when you save the job definition. To determine which overrides were applied after a job run, you can reference the over_num value. For example, when applying a job override, the scheduler specifies the override it is using as follows:

```
Job: JOB_NAME is using Over-Ride #14
```

Note: You cannot edit a job override specified using jil. If you specify an override for a job, but an override already exists, the new override replaces the original one. However, the original override (the over_num value) is maintained in the database.

Syntax

This subcommand has the following formats:

- To override one or more attributes:

```
override_job: job_name
attribute_keyword: value | NULL
attribute_keyword: value | NULL
```

·
·
·

- To delete an override:

```
override_job: job_name delete
```

job_name

Specifies a job or Box job that is defined in the database.

delete

Cancels the previously specified job override for *job_name*.

attribute_keyword

Defines the job attribute to override. You can specify one of the following:

- | | |
|--|--|
| <ul style="list-style-type: none">■ auto_hold■ command■ condition■ date_conditions■ days_of_week■ exclude_calendar■ machine■ max_run_alarm■ min_run_alarm■ n_retrys■ profile | <ul style="list-style-type: none">■ run_calendar■ run_window■ start_mins■ start_times■ std_err_file■ std_in_file■ std_out_file■ term_run_time■ watch_file■ watch_file_min_size■ watch_interval |
|--|--|

value

Defines the value to override the specified attribute with.

NULL

Deletes or negates the current value of the specified attribute.

Example: Define a One-time Job Override

This example defines a one-time job override to change the standard output file for the job, JobA.

```
override_job: JobA  
std_out_file: "C:\OUTPUT\SpecialRun.out"
```

Example: Cancel a Job Override

This example cancels the job override for JobA.

```
override_job: JobA delete
```

update_job Subcommand—Update an Existing Box or Job Definition in the Database

The update_job subcommand updates an existing box or job definition in the database. The update_job statement is followed by a list of *attribute:value* statements.

Attributes in the existing definition that are not explicitly replaced in the update_job input retain their original settings.

Note: When many attributes must be updated for a job, it may be more efficient to delete and re-insert the job definitions.

Syntax

This subcommand has the following format:

```
update_job: job_name  
job_name
```

Specifies the name of the job whose definition you want to update.

Limits: Up to 64 characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (_), hyphen (-), and pound (#); do not include embedded spaces or tabs

Example: Update a Command

This example changes the preexisting command job time_stamp to run on the machine paris instead of on the originally specified machine:

```
update_job: time_stamp  
machine: paris
```

Common Job Attributes

Some JIL attributes are common to all job types. For example, you can define any job to send an alarm if the job fails or terminates. You can also define starting or restart conditions for any job.

Required Attributes for All Job Types

The following attribute is required for all job types:

- machine

Note: By default, the job type is set to CMD. You can specify a different job type using the job_type attribute.

Optional Attributes for All Job Types

The following attributes are optional for all job types:

- alarm_if_fail
- application
- auto_delete
- auto_hold
- avg_runtime
- box_name
- box_terminator
- condition
- date_conditions
- days_of_week
- description
- exclude_calendar
- group
- job_load
- job_type
- max_run_alarm
- min_run_alarm
- must_complete_times
- must_start_times
- n_retrys
- notification_id
- notification_msg
- owner (This attribute does not apply to File Trigger jobs.)
- permission
- priority
- resources
- run_calendar
- run_window
- send_notification
- service_desk
- start_mins
- start_times
- svcdesk_attr
- svcdesk_desc
- svcdesk_imp
- svcdesk_pri
- svcdesk_sev
- term_run_time
- timezone

alarm_if_fail Attribute—Specify Whether to Post an Alarm for FAILURE or TERMINATED Status

The alarm_if_fail attribute specifies whether to post an alarm to the scheduler to alert the operator to take corrective action when the job completes with a FAILURE or TERMINATED status.

Supported Job Types

This attribute is optional for all job types.

Syntax

This attribute has the following format:

`alarm_if_fail: y | n`

y

Posts an alarm to the scheduler when the job fails or is terminated.

Note: You can specify 1 instead of y.

n

Does not post an alarm to the scheduler when the job fails or is terminated. This is the default.

Note: You can specify 0 instead of n.

Example: Generate an Alarm when a Job Fails or is Terminated

This example generates an alarm when the job fails or is terminated:

`alarm_if_fail: y`

application Attribute—Associate a Job with an Application

The application attribute associates a job with a specific application so users can classify, sort, and filter jobs by application name.

You can use the application attribute with the sendevent, job_depends, and autorep commands. For example, you could use a single sendevent command to start all of the jobs associated with a specific application.

Consider the following when you use the application attribute:

- Users with execute permission can add new application attributes to a job definition.
- You can only associate a job with one application at a time.
- When a job is associated with a group (using the group attribute) and an application, the product assumes that the application attribute is a subset of the group attribute.

Supported Job Types

This attribute is optional for all job types.

Syntax

This attribute has the following format:

`application: application_name`

application_name

Defines the name of the application with which to associate the job. You can only associate a job with one application, but an application may have many jobs associated with it.

Limits: Up to 64 characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (_), pound (#), and hyphen (-); do not include embedded spaces or tabs

Example: Associate a Job with an Application

This example associates a job with the application summary_reports:

`application: summary_reports`

arc_obj_name Attribute—Identify Name of SAP Archiving Object

The arc_obj_name attribute identifies the name of an archiving object defined on the SAP system.

Supported Job Type

This attribute is required for the [SAP Data Archiving \(SAPDA\) job type](#) (see page 259).

Syntax

This attribute has the following format:

arc_obj_name: *object_name*

object_name

Specifies the name of the archiving object.

Limits: Up to 256 characters; case-sensitive

Example: Specify an Archiving Object

This example defines a job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
insert_job: SAPDA_job
job_type: SAPDA
machine: sapagent
owner: WAAESAP@sapagent
arc_obj_name: BC_ARCHIVE
arc_obj_variant: BC_ARCVARIANT
sap_print_parms: dest=LP01,prt_arc_mode=PRINT
```

arc_obj_variant Attribute—Identify Name of SAP Archiving Object Variant

The arc_obj_variant attribute identifies the name of an archiving object variant defined on the SAP system.

Supported Job Type

This attribute is required for the [SAP Data Archiving \(SAPDA\) job type](#) (see page 259).

Syntax

This attribute has the following format:

arc_obj_variant: *object_variant*

object_variant

Specifies the name of the archiving object variant.

Limits: Up to 256 characters; case-sensitive

Example: Specify an Archiving Object Variant

This example defines a job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
insert_job: SAPDA_job
job_type: SAPDA
machine: sapagent
owner: WAAESAP@sapagent
arc_obj_name: BC_ARCHIVE
arc_obj_variant: BC_ARCVARIANT
sap_print_parms: dest=LP01,prt_arc_mode=PRINT
```

arc_parms Attribute—Specify Archive Parameters

The arc_parms attribute specifies the archive parameters for an SAP Data Archiving object.

Note: To use this attribute, you must specify prc_arc_mode=ARCHIVE or prc_arc_mode=BOTH in the sap_print_parms attribute.

Supported Job Type

This attribute is optional for the [SAP Data Archiving \(SAPDA\) job type](#) (see page 259).

Syntax

This attribute has the following format:

```
arc_parms: arc_doc_type=value,arc_info=value,arc_obj_type=value  
[,keyword=value...]
```

arc_doc_type=value

Specifies the SAP ArchiveLink type. This keyword corresponds to the Doc type field on the Define Background Archive Parameters dialog.

Limits: Up to 10 characters; case-sensitive

Example: archive

Note: This keyword is required if prc_arc_mode is set to BOTH or ARCHIVE.

arc_info=value

Specifies the SAP ArchiveLink information.

Limits: Up to three characters; case-sensitive

Example: inf

Note: This keyword is required if prc_arc_mode is set to BOTH or ARCHIVE.

arc_obj_type=value

Specifies the type of external system archive object. This keyword corresponds to the Obj. type field on the Define Background Archive Parameters dialog.

Limits: Up to 10 characters; case-sensitive

Example: archive

Note: This keyword is required if prc_arc_mode is set to BOTH or ARCHIVE.

keyword=value

(Optional) Specifies an archive parameter. You can specify multiple parameters. Separate each keyword=value pair with a comma. Options include the following:

arc_client=value

Specifies the archive link client.

Limits: Up to 3 digits

Example: 800

arc_connect=value

Specifies the archive link communication connection.

Limits: Up to 14 characters; case-sensitive

Example: arconnect

arc_date=value

Specifies the archive link archiving date. The format is *yyyymmdd*.

Limits: Up to eight numeric digits

Example: 20080702

arc_doc_class=value

Specifies the SAP ArchiveLink document class.

Limits: Up to 20 characters; case-sensitive

Example: arcdocclass

arc_format=value

Specifies the SAP ArchiveLink output format.

Limits: Up to 16 characters; case-sensitive

Example: X_65_132

arc_host_link=value

Specifies the SAP ArchiveLink RPC host.

Limits: Up to 32 characters; case-sensitive

Example: rpchost

arc_path=value

Specifies the standard archive path.

Limits: Up to 70 characters; case-sensitive

Example: /export/home/archive

arc_printer=*value*

Specifies the SAP ArchiveLink target printer.

Limits: Up to 4 characters; case-sensitive

Example: LP01

arc_protocol=*value*

Specifies the archive storage connection protocol.

Limits: Up to eight characters; case-sensitive

Example: prtcl12

arc_report=*value*

Specifies the report name.

Limits: Up to 30 characters; case-sensitive

Note: Enclose values that contain spaces in quotation marks ("").

Example: "Archive Report"

arc_service=*value*

Specifies the RPC service/RFC destination.

Limits: Up to 32 characters; case-sensitive

Example: rpcdest

arc_storage=*value*

Specifies the SAP ArchiveLink target storage system.

Limits: Up to two characters; case-sensitive

Example: st

arc_text=*value*

Specifies the archive link Text Information field.

Limits: Up to 30 characters; case-sensitive

Note: Enclose values that contain spaces in quotation marks ("").

Example: "Archive text"

arc_userid

Specifies the archive data element for user.

Limits: Up to 12 characters

Example: arcuser

arc_version=value

Specifies the archive version.

Limits: Up to four characters

Example: 4.1

Note: The entire value can be up to 4096 characters.

Example: Specify Archive Parameters

This example specifies the archive parameters in an SAP Data Archiving job.

```
insert_job: test_SAPDA
job_class: SAP
job_type: SAPDA
machine: localhost
owner: WAAESAP@waae-jobstest
arc_obj_name: BC_ARCHIVE
arc_obj_variant: PRODUKTIVMODUS
sap_print_parms: prt_arc_mode=BOTH,dest=LP01,title="testing"
arc_parms: arc_doc_type=ar01, arc_info=ar1, arc_obj_type=obj01, arc_client=202
```

auth_string Attribute—Define an Authorization String

The auth_string attribute defines the authorization string the scheduling manager uses to post the correct job. The scheduling manager posts the manual job complete when the manually submitted job runs under the corresponding user ID.

Supported Job Type

This attribute is optional for the [z/OS Manual \(ZOSM\) job type](#) (see page 284).

Syntax

This attribute has the following format:

```
auth_string: string  
string
```

Defines the authorization string the scheduling manager uses to post and track the correct job.

Limits: Up to 80 characters

Note: Contact your agent administrator to confirm whether the agent checks the authorization string. Depending on the scenario, the action to take differs as follows:

- If authorization checking (AUTHSTR) is not set up for this agent, do not specify the authorization string.
- If authorization checking (AUTHSTR) is set to RACUSER, specify the user ID that owns the job.
- If authorization checking (AUTHSTR) is set to ACCOUNT1, ACCOUNT2, ACCOUNT3, or ACCOUNT4, specify the value found in the corresponding position of the job's Job Control Language (JCL). For example, if AUTHSTR is set to check the ACCOUNT2 field, specify the value found in the second accounting position in the job's JCL.

Example: Post a z/OS Manual Job as Complete Based on the User ID

This example posts a z/OS Manual job as complete when the manually-submitted job ABC runs under user CYBER. The ZOS1 agent monitors job ABC.

```
insert_job: ABC_job  
job_type: ZOSM  
machine: ZOS1  
zos_jobname: ABC  
owner: zosuser  
auth_string: CYBER
```

auto_delete Attribute—Automatically Delete a Job on Completion

CA Workload Automation AE can automatically delete a job after it completes. The auto_delete attribute specifies the number of hours to wait after a job completes. If the job is a box, the box and all the jobs it contains are deleted. This attribute is useful for scheduling and running a one-time batch job.

Supported Job Types

This attribute is optional for all job types.

Syntax

This attribute has the following format:

auto_delete: *hours* | 0 | -1

hours

Defines the number of hours to wait after the job completes. At that time, the job is automatically deleted.

Limits: 1-17520

0

Deletes the job immediately after successful completion. If the job does not complete successfully, the job definition is kept for seven days before it is automatically deleted.

-1

Does not automatically delete the job. This is the default.

Example: Delete a Job on Successful Completion

This example automatically deletes the job five hours after the job completes successfully.

```
insert_job: unix_script
job_type: CMD
machine: unixagent
command: /usr/common/backup
auto_delete: 5
```

auto_hold Attribute—Put a Job on Hold when Its Container’s Status Changes to RUNNING

When a job is in a box, it inherits the starting conditions of the box. When the box’s status changes to RUNNING, all the jobs in the box also start unless other conditions are not satisfied.

The auto_hold attribute specifies whether to change the state of a job that is in a box to ON_HOLD when the status of the containing box changes to RUNNING.

Supported Job Types

This attribute is optional for all job types. This attribute only applies to jobs that are in a box.

Syntax

This attribute has the following format:

auto_hold: y | n

y

Automatically changes the state of the job to ON_HOLD when its containing box starts running.

Note: You can specify 1 instead of y.

n

Does not put the job on hold when its containing box starts running. This is the default.

Note: You can specify 0 instead of n.

Note: To take a job off hold, use the sendevent command to send the JOB_OFF_HOLD event for the job.

Example: Put a Job on Hold when its Container’s Status Changes to RUNNING

This example defines a box that contains two jobs. When the status of the box changes to RUNNING, the autonoHold job runs, but the autohold job does not run because it is on hold.

```
insert_job: holdBox
job_type: BOX
owner: root@localhost

insert_job: autohold
job_type: CMD
box_name: holdBox
command: ls
```

```
machine: localhost
owner: root@localhost
auto_hold: y

insert_job: autonoHold
job_type: CMD
box_name: holdBox
command: ls
machine: localhost
owner: root@localhost
```

avg_runtime Attribute—Define Average Run Time for a New Job

The avg_runtime attribute defines an average run time (in minutes) for a job that is newly submitted to the database. The attribute is used when the job has not run.

Supported Job Types

This attribute is optional for all job types.

Syntax

This attribute has the following format:

```
avg_runtime: value
value
```

Defines the average run time (in minutes) to use for a job that has not run.

Default: 0

Limits: 527,040

Note: When the DBMaint script runs, it recalculates the average run time for each job in the database.

Windows Note: CA Workload Automation AE only uses the avg_runtime attribute to establish an average run time for a new job in the ujo_avg_job_runs table.

Example: Define Average Run Time for a New Job

This example sets the average run time for a new job to five minutes.

```
avg_runtime: 5
```

bdc_err_rate Attribute—Specify the Maximum Acceptable Error Rate

The bdc_err_rate attribute specifies the maximum acceptable error rate in an SAP Batch Input Session job.

Supported Job Type

This attribute is optional for the [SAP Batch Input Session \(SAPBDC\) job type](#) (see page 255).

Syntax

This attribute has the following format:

`bdc_err_rate: percent`

percent

Specifies the maximum acceptable error rate as a percentage of the total transactions. When set to 100, all errors are acceptable. When set to zero (0), the job fails if any transaction contains an error. When set to 5, for example, the job fails if more than five percent of the transactions contain errors.

Limits: 0-100

Example: Specify a Maximum Acceptable Error Rate

In this example, the maximum acceptable error rate is five percent. If more than five percent of transactions contain errors, the job fails.

```
insert_job: test_SAPBDC
job_type: SAPBDC
machine: sapagent
owner: WAAESAP@waae-jobstest
sap_job_name: ZBDCTEST
sap_release_option: A
sap_step_parms: abap_name=ZBDCTEST
bdc_err_rate: 5
```

bdc_ext_log Attribute—Generate Advanced Logging of the Batch Input Session

The bdc_ext_log attribute specifies whether to generate advanced logging of the Batch Input Session (BDC) running on the SAP system.

Note: If you do not specify the bdc_ext_log attribute in your job definition, the job does not generate advanced logging (the default).

Supported Job Type

This attribute is optional for the [SAP Batch Input Session \(SAPBDC\) job type](#) (see page 255).

Syntax

This attribute has the following format:

bdc_ext_log: Y | N

Y

Generates advanced logging.

N

Does not generate advanced logging. This is the default.

Example: Generate Advanced Logging of the Batch Input Session

This example generates advanced logging of the BDC running on the SAP system.

```
insert_job: test_SAPBDC
job_type: SAPBDC
machine: sapagent
owner: WAAESAP@waae-jobstest
sap_job_name: ZBDCTEST
sap_release_option: A
sap_step_parms: abap_name=ZBDCTEST
bdc_ext_log: Y
```

bdc_proc_rate Attribute—Specify Minimum Acceptable Process Rate

The bdc_proc_rate attribute specifies the minimum acceptable process rate within an SAP Batch Input Session job.

Supported Job Type

This attribute is optional for the [SAP Batch Input Session \(SAPBDC\) job type](#) (see page 255).

Syntax

This attribute has the following format:

`bdc_proc_rate: percent`

percent

Specifies the minimum acceptable process rate as a percentage of the total transactions. When set to 100, the job fails if all transactions are not processed. When set to zero (0), no minimum processed transactions are required. When set to 90, for example, the job fails if less than 90 percent of the transactions are processed.

Limits: 0-100

Example: Specify a Minimum Acceptable Process Rate

In this example, the minimum acceptable process rate is 90 percent. If less than 90 percent of transactions are processed, the job fails.

```
insert_job: test_SAPBDC11_SIMPLE
job_type: SAPBDC
machine: sapagent
owner: WAAESAP@waae-jobstest
permission:
date_conditions: n
alarm_if_fail: y
sap_job_name: ZBDCTEST
sap_office: N
sap_release_option: I
bdc_ext_log: N
bdc_proc_rate: 90
sap_step_parms:
abap_name="ZBDCTEST",banner_page=N,release=N,print_imm=N,new_spool=N,footer=N
```

bdc_system Attribute—Specify the Name of a Background Server

The bdc_system attribute specifies the name of the background server that processes the Batch Input Session (BDC) running on the SAP system.

Supported Job Type

This attribute is optional for the [SAP Batch Input Session \(SAPBDC\) job type](#) (see page 255).

Syntax

This attribute has the following format:

`bdc_system: destination`

destination

Specifies the name of the background server.

Limits: Up to 256 valid SAP characters; case-sensitive

Example: Specify the Background Processing Server Name

This example defines a job that uses a background server named dest to process the Batch Input Session running on the SAP system.

```
insert_job: test_SAPBDC11_SIMPLE2
job_type: SAPBDC
machine: sap_agent
owner: WAAESAP@waae-jobtest
sap_job_name: ZBDCTEST
sap_office: N
sap_release_option: I
bdc_ext_log: N
sap_step_parms:
abap_name="ZBDCTEST",banner_page=N,release=N,print_imm=N,new_spool=N,footer=N
bdc_system:dest
```

bean_name Attribute—Specify the JNDI Name of the Bean

The bean_name attribute specifies the JNDI name of the bean in an Entity Bean and Session Bean job.

Supported Job Types

This attribute is required for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

Syntax

This attribute has the following format:

bean_name: *bean*

bean

Specifies the JNDI name of the session or entity bean.

Limits: Up to 256 characters; case-sensitive

Example: Invoke a Method on a Stateless Session Bean

Suppose that you want to invoke the reverse method on the CybEJBTestBean stateless session bean. The reverse method has one parameter, with type java.lang.String and value a23. The output from the reverse method is saved in the C:\Makapt15 file. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere application server and 2809 is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

blob_file Attribute—Specify File for the Blob

The blob_file attribute specifies the file containing the data that is used to create an input job blob. When you specify this attribute in a job definition, the blob is uploaded to the database when the job is defined. This input job blob is associated with the job. The job itself can then use the blob as input at run time. Other jobs that run on different computers can also access this blob by specifying the appropriate keyword in the std_in_file attribute.

Note: Alternatively, you can create an input job blob after a job is defined by using the insert_blob subcommand.

Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

Syntax

This attribute has the following format:

```
blob_file: file  
file
```

Specifies the file containing the data to insert for the blob.

Limits: Up to 255 characters

Examples:

- /tmp/blob_input
- "c:\temp\blob_input"

Example: Create an Input Job Blob Using a File

This example defines a CMD job that creates and uses an input job blob. When the job is defined, a blob is created using the textual data contained in the blob_input_file.txt file. The blob is associated with the job. When the job runs, it uses the created blob as input and treats the blob data as textual data.

```
insert_job: test_blob  
job_type: CMD  
machine: unixagent  
command: cat  
blob_file: /blob_input_file.txt  
std_in_file: $$blobt  
owner: produser@unixagent
```

More information:

[std_in_file Attribute—Redirect the Standard Input File](#) (see page 683)

blob_input Attribute—Specify Input for the Blob

The blob_input attribute specifies one or more lines of text to contain in a blob. Unlike the blob_file attribute that lets you specify a file that contains the data to store in the blob, the blob_input attribute lets you type the blob's data directly in the job definition.

When you specify the blob_input attribute in a job definition, the blob is uploaded to the database when the job is defined. This input job blob is associated with the job. The job itself can then use the blob as input at run time. Other jobs that run on different computers can also access this blob by specifying the appropriate keyword in the std_in_file attribute.

Note: Alternatively, you can create an input job blob after a job is defined by using the insert_blob subcommand.

Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

Syntax

This attribute has the following format:

`blob_input: <auto_blob>input</auto_blob>`

input

Specifies the text to insert for the blob.

Note: You can specify multiple lines of text. All text enclosed in the auto_blob XML-style metatags is stored literally in the database.

Example: Create an Input Job Blob

This example defines a CMD job that creates and uses an input job blob. When the job is defined, a blob is created using the text that is specified in the blob_input attribute. When the job runs, it uses the created blob as input and treats the blob data as textual data.

```
insert_job: test_blob
job_type: CMD
command: cat
machine: unixagent
blob_input: <auto_blobt>multi-lined text data for job blob
</auto_blobt>
std_in_file: $$blobt
owner: produser@unixagent
```

More Information:

[std_in_file Attribute—Redirect the Standard Input File](#) (see page 683)

box_failure Attribute—Define Criteria for Box Job Failure

The box_failure attribute defines dependencies that indicate when a Box job has not completed successfully. The box_failure condition is in addition to the status of each job in the box.

By default, CA Workload Automation AE considers a Box job to have failed when any job in it completes with a failure condition. However, a Box job can contain complex logic which can take a number of different paths, any of which may constitute failure or one of which can include recovery from a failed job. You can use the box_failure attribute to define specific conditions for Box job failure in situations where there may be multiple failure or recovery scenarios. Jobs within the box that fail still result in the box being marked as failed. The box_failure condition allows jobs outside the box to be included in determining the box status in addition to other conditions such as global variables.

You can base Box job failure on one or more of the following:

- Job status; for example, failure(DB_BACKUP)
- Job status across instances; for example, failure(jobB^PRD)
- Jobs status with look-back; for example, success (job1,01.00)
- Job exit code; for example, exitcode (my_job)=4
- Global variables; for example, VALUE(TODAY)=Friday

Support Job Types

This attribute applies only to BOX job types.

Syntax

This attribute has the following format:

`box_failure: conditions`

conditions

Specifies the job failure conditions using a logical expression. The Box job is evaluated against these conditions in addition to all job statuses of jobs within the box. If no job failure condition is specified, the Box job fails if at least one job in the box fails.

Limits: Up to 4096 characters

Keep the following in mind when defining conditions for Box job failure:

- You can use the pipe symbol (|) instead of the word OR, and the ampersand symbol (&) instead of the word AND.
- Spaces between conditions and delimiters are optional.
- You can specify even more complex conditions by grouping the expressions in parentheses. The parentheses force precedence, and the equation is evaluated from left to right. For example, the following attribute definition indicates that the box job is considered to have failed when either JobA and JobB return FAILURE or JobD returns FAILURE (for example, if you set up JobD to attempt to recover a failed job):

`box_failure: (failure(JobA) AND failure(JobB)) OR failure(JobD)`

Example: Fail a Box Job when JobA or JobB but Not JobC Fails

This example sets the status of the Box job to FAILURE when JobA or JobB fails, but not when JobC fails:

`box_failure: failure(JobA) OR failure(JobB)`

Example: Fail a Box Job when JobA, JobB, and JobC Fail

This example sets the status of the Box job to FAILURE only when all three jobs (JobA, JobB, and JobC) fail:

`box_failure: failure(JobA) AND failure(JobB) AND failure(JobC)`

More information:

[Job Status Dependencies](#) (see page 337)

[Cross-Instance Job Dependencies](#) (see page 340)

[Look-Back Dependencies](#) (see page 344)

[Exit Code Dependencies](#) (see page 342)

[Global Variable Dependencies](#) (see page 343)

box_name Attribute—Identify a Box as Container for a Job

The box_name attribute identifies an existing box job to put a job in. Boxes let you process a set of jobs as a group. This is useful for setting starting conditions at the box level to “gate” the jobs in the box. You can specify each job’s starting conditions relative to the other jobs in the box as appropriate.

Supported Job Types

This attribute is optional for all job types.

Syntax

This attribute has the following format:

box_name: *name*

name

Specifies an existing box job to put the job in.

Limits: Up to 64 characters

Example: Identify a Box as Container for a Job

This example puts the job you are defining in an existing box job named Box_1:

box_name: Box_1

box_success Attribute—Define Criteria for Box Job Success

The box_success attribute defines which of the dependencies specified in the condition attribute should indicate when a box job has completed successfully.

By default, CA Workload Automation AE considers a box job successful when every job in it completes successfully. However, a box job can contain complex logic which can take a number of different paths, any of which may constitute success. In such cases, some jobs in the box may never need to run, but the default box behavior would be to interpret this as a box failure. You can use the box_success attribute to define specific conditions for box job success in situations where there may be multiple success scenarios.

You can base box job success on one or more of the following:

- Job status; for example, success(DB_BACKUP)
- Job status across instances; for example, success(jobB^PRD)
- Jobs status with look-back; for example, success (job1,01.00)
- Job exit code; for example, exitcode (my_job)=4
- Global variables; for example, VALUE(TODAY)=Friday

Supported Job Types

This attribute applies only to BOX job types and is optional.

Syntax

This attribute has the following format:

box_success: *conditions*

conditions

Specifies the job success conditions using a logical expression. The Box job is evaluated against these conditions. If no job success conditions are specified, the Box job succeeds only if all the jobs in the box complete successfully.

Limits: Up to 4096 characters

Keep the following in mind when defining conditions for box job success:

- You can use the pipe symbol (|) instead of the word OR, and the ampersand symbol (&) instead of the word AND.
- Spaces between conditions and delimiters are optional.

- You can specify even more complex conditions by grouping the expressions in parentheses. The parentheses force precedence, and the equation is evaluated from left to right. For example, the following attribute definition indicates that the box job is considered successful when either JobA and JobB return SUCCESS or JobD and JobE return SUCCESS, FAILURE, or TERMINATED:

```
box_success: (success(JobA) AND success(JobB)) OR (done(JobD)  
AND done(Job E))
```

Note: In multiple conditions (that is, if JobC is dependent on SUCCESS of JobA *and* SUCCESS of JobB), JobC runs when both JobA and JobB succeed, no matter when the first SUCCESS event occurs. This may not be the desired result, for example, for a daily processing cycle where JobA finished yesterday but did not run today, and JobB succeeded today. To avoid this result, you should group the jobs in a box.

[Example: Set a Box Job to SUCCESS when JobA or JobB Succeeds](#)

This example sets the status of the box job to SUCCESS when JobA succeeds or JobB succeeds, but ignores the status of JobC:

```
box_success: success(JobA) OR success(JobB)
```

[Example: Set a Box Job to SUCCESS when JobA and JobB Complete Successfully, and JobC Completes](#)

This example sets the status of the box job to SUCCESS when jobs JobA and JobB succeed and JobC completes (regardless of its status):

```
box_success: success(JobA) AND success(JobB) AND done(JobC)
```

More information:

[Job Status Dependencies](#) (see page 337)
[Cross-Instance Job Dependencies](#) (see page 340)
[Look-Back Dependencies](#) (see page 344)
[Exit Code Dependencies](#) (see page 342)
[Global Variable Dependencies](#) (see page 343)

box_terminator Attribute—Terminate Box on Job Failure

The box_terminator attribute specifies whether to terminate the box job containing the job you are defining when the job completes with a FAILURE or TERMINATED status.

Use this attribute with the job_terminator attribute to control how nested jobs react when a Box job fails.

Supported Job Types

This attribute applies only to BOX job types and is optional.

Syntax

This attribute has the following format:

box_terminator: y | n

y

Terminates the Box job that contains the job you are defining when the job fails or terminates.

Note: You can specify 1 instead of y.

n

Does not terminate the Box job that contains the job. This is the default.

Note: You can specify 0 instead of n.

Example: Terminate Box on Job Failure

This example terminates the containing Box job if the job you are defining fails or is terminated.

```
box_terminator: y
```

chk_files Attribute—Verify File Space Required to Start a Job

The chk_files attribute defines the minimum amount of file space that must be available on the specified UNIX file system or Windows drive before the job can start. At run time, the agent checks whether the required space is available on the machine where the job runs. If the requirements are met, the job starts. If the requirements are not met, the agent generates an alarm and the job does not start. The system tries to verify the file space again and start the job. The number of tries is determined using the n_retrys attribute. If the n_retrys attribute is not specified in the job definition, the number of tries is determined by the MaxRestartTrys parameter in the configuration file (UNIX) or the Max Restart Trys field in the Administrator utility (Windows). If the required space is still not available after all the restart attempts, the job fails.

By default, the available file space is not checked.

Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

Syntax

This attribute has the following format:

chk_files: *location size* [*location size...*]

location

Specifies a Windows drive or the full path name to a UNIX file system where the file space is required.

Note: You can specify multiple location and size pairs. Specify each pair with a space.

UNIX: You can specify the full path to a directory in a file system.

Windows: Only drives are checked. If directories are specified, they are ignored.

size

Specifies the required amount of file space. This value must be less than the amount of available space on the file system or drive. Otherwise, the job fails. Optionally, you can specify the size qualifier. Options are the following:

B

Specifies that the value is in bytes.

KB

Specifies that the value is in kilobytes. This is the default.

M

Specifies that the value is in megabytes.

G

Specifies that the value is in gigabytes.

Limits: Any integer

Examples: 200, 20G

Note: The entire value can be up to 255 characters.

Example: Verify the Available File Space on UNIX

This example checks whether the file system named roots has 100 KB of available space. This example also checks whether the file system named auxfs1 has 120 KB of available space. The specified file space must be available before the job can start.

```
insert_job: unix_chk
job_type: CMD
machine: unixagent
command: /u1/procrun.sh
chk_files: /roots 100 /auxfs1 120
```

Example: Verify the Available File Space on Windows

This example checks whether the C: drive has 100 KB of available space and the D: drive has 120 KB of available space. The specified file space must be available before the job can start.

```
insert_job: win_chk
job_type: CMD
machine: winagent
command: "C:\Programs\Payroll\pay.exe"
chk_files: "C: 100 D: 120"
```

class_name Attribute—Specify a Class Name

The class_name attribute specifies the name of the Java class to instantiate in a POJO job or the fully qualified Java class of the MBean object in a JMX-MBean Create Instance job.

Supported Job Types

This attribute is required for the following job types:

- [JMX-MBean Create Instance \(JMXMLC\)](#) (see page 235)
- [POJO](#) (see page 252)

Syntax

This attribute has the following format:

class_name: *class*
class

Specifies the Java class to instantiate in a POJO job or the fully qualified Java class of the MBean object in a JMX-MBean Create Instance job.

Limits: Up to 1024 characters; case-sensitive

Example: Specify Java Class to Instantiate in a POJO Job

Suppose that you want to define a POJO job that creates a Java String with value "5" and calls the parseInt method with the created Java String object as an argument. The parseInt method returns a Java Integer object.

```
insert_job: ignore
job_type: POJO
machine: appagent
class_name: java.lang.Integer
method_name: parseInt
j2ee_parameter: java.lang.String=5
```

Example: Specify Java Class of the MBean Object

Suppose that you want to create an MBean instance on a JMX server. The job uses the cdc.jmx.SimpleDynamic class. The constructor of the class takes a single string parameter with the value "Hello".

```
insert_job: create
job_type: JMXMC
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic"
class_name: cdc.jmx.SimpleDynamic
jmx_parameter: java.lang.String="Hello"
```

command Attribute—Specify a Command or Script to Run

The command attribute specifies a command, executable, UNIX shell script, application, or batch file to run when all the starting conditions are met. The attribute can also specify arguments to pass to the command.

Supported Job Type

This attribute is required for the [Command \(CMD\) job type](#) (see page 204).

Syntax

This attribute has the following format:

`command: command argument...`

command

Specifies the command, executable, UNIX shell script, application, or batch file to run when all the starting conditions are met.

Note: Enclose values that contain embedded blanks in quotation marks.

Example: “C:\Program Files\Application\program.exe”

argument...

Specifies one or more arguments to pass to the command or script at run time.

Note: Separate each argument with a space. You must specify each argument in the order it is expected in the program. Single arguments that contain embedded blanks must be delimited by quotes.

Example: “a value”

Notes:

- The entire value can be up to 512 characters.
- When issuing commands or defining jobs that run on a different operating system (for example, Windows to UNIX or UNIX to Windows), you must use the syntax appropriate to the operating system where the job will run.
- The specified command, batch file, UNIX shell script, or executable does not have to exist at job definition time. However, it must exist at run time.
- Global variables (set using the sendevent command) are valid in the command name or in the command's run-time arguments. You can use global variables to pass command line arguments. You can use one of the following formats to reference global variables:
 - `$$global_name`
 - `$$\{global_name\}`
- Global variable names can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (_), hyphen (-), and pound (#). Global variable names can also contain spaces.

UNIX Considerations for the command Attribute

When you define a CMD job to run a UNIX script or command, consider the following points when specifying the command attribute:

Permission level of the job owner:

The job owner must have execute permission for the command on the client.

Syntax restrictions:

- Because CA Workload Automation AE performs an exec to run the command, you cannot separate multiple commands with semi-colons.
- UNIX systems are case-sensitive. The specified path and script name must match the path and script name on the UNIX system or the job will fail.
- Piping or redirection of standard input, output, and error files is not allowed. Use shell scripts to execute piped commands and attributes (such as std_in_file for standard input) to provide the necessary functionality.

Running a C script:

If you are running a C-Shell (csh) script, the system attempts to source a .cshrc file when it begins interpreting the file. Although this may be preferred, the system also overwrites variables defined in the profile script. If you do not want to have the .cshrc file sourced, you must invoke the csh script with the -f option. In this case, the first line of the script resembles the following:

```
#!/bin/csh -f
```

Using environment variables that are defined in the job profile:

- The command specified in the command attribute runs in the environment defined in the /etc/auto.profile file, the job profile, and the envvars attribute. The job profile is a shell script and can be specified in the job definition using the profile attribute. For more information about the job profile, see the profile attribute and the *User Guide*.
- If \$PATH is assigned in the job profile, CA Workload Automation AE searches that path to find the command. When you specify the full path name, you can use variables exported from the profile script in the path name specification. If you use variable substitution, enclose the variable in braces ({ }). For example, you can specify the following:

`${PATH}`

- You can define the entire environment needed for the command in the profile that will be sourced. Alternatively, you can define environment variables at the job level using the envvars attribute.
- If a command works properly when issued at a shell prompt, but fails to run properly when specified as a command attribute, the shell and CA Workload Automation AE environments are probably different. In this situation, check that all required command variables are specified in the job profile or envvars attribute.

Specifying the command or script name without the full path:

The command attribute usually requires the full path to the command or script name you want to run. However, you can specify the name without the full path if all of the following conditions are true:

- The agent is running under the root account.
- The agent is configured to resolve environment variables.
- Using the owner attribute, you specify a user ID that has the authority to run the job on the agent computer. The agent uses the user default shell.
- The path to the command name is set in the PATH system environment variable for the specified user ID.
- The oscomponent.lookupcommand=true parameter is set in the agentparm.txt file of the agent.

Specifying a script that calls a second script:

When you define a job that runs a script that calls a second script, the fully qualified path of the called script must be provided.

Specifying the command script name using an environment variable:

- You can specify the command or script name using an environment variable, for example \$MY_PATH/myscript.sh, if all of the following conditions are true:
 - The agent is running under the root account.
 - The agent is configured to resolve environment variables.
 - Using the owner attribute, you specify a user ID that has the authority to run the job on the agent computer. The agent uses the user default shell.
 - The environment variable used, for example \$MY_PATH, is set in the specified user ID's profile file.

Determining which shell is used to run the UNIX script:

The shell can be specified in different places. To run a UNIX script, the agent uses, in the following order, the shell specified in the following places:

1. The shell attribute (if specified in the job definition)
2. The first line of the script (if the shell attribute is not specified)
3. The oscomponent.defaultshell parameter in the agentparm.txt file (if not specified in the shell attribute or in the script)
4. The user default shell defined in the user profile (if not specified in one of the previous three locations)

Note: If the oscomponent.checkvalidshell parameter in the agent's agentparm.txt file is set to true (the default), all shells that are used must be specified using the oscomponent.validshell parameter. The path defined in the first line of the script or in the job definition must match the corresponding path in the oscomponent.validshell parameter. If the shell you want to use is not defined on the agent, the job fails. For more information about specifying valid shells, see the oscomponent.checkvalidshell and oscomponent.validshell parameters in the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

Examples: Specifying the command Attribute on UNIX

The following examples are Command Jobs on UNIX:

Example: Run a Command on UNIX

This example runs the UNIX date command on the UNIX machine named unixagent. If the /bin directory is included in the search path, either in the /etc/ auto.profile or in the user-defined profile, you do not have to specify the full path to the command. Instead, you can specify the command name only (for example, command: date).

```
insert_job: unix_cmd
job_type: CMD
machine: unixagent
command: /bin/date
```

Example: Run a Script on UNIX

This example runs the Backup script that is located in the /usr/common directory. The job runs on the UNIX machine named unixagent. If the /usr/common directory is included in the job's runtime environment path, you do not have to specify the full path to the script. Instead, you can specify the script name only (for example, command: backup)

```
insert_job: unix_script
job_type: CMD
machine: unixagent
command: /usr/common/backup
```

Example: Remove All Files from a Subdirectory on UNIX

This example removes all files from the tmp subdirectory on the unixagent machine. The tmp subdirectory is located in the directory specified in the MY_BACKUPS global variable.

```
insert_job: unix_remove
job_type: CMD
machine: unixagent
command: rm $$\{MY_BACKUPS\}/tmp/*
```

Example: Redirect I/O and Pipe Output on UNIX

This example redirects I/O and pipes output from a command. The agent runs the UNIX command ps -ef|grep someValue and pipes output to the /tmp/log file. The shell interpreter is invoked in the command attribute. The command values are passed as positional parameters to the interpreter.

```
insert_job: unix_job
job_type: CMD
machine: unixagent
command: /usr/bin/ksh -c "ps -ef|grep someValue >>/tmp/log"
owner: user1
```

Windows Considerations for the command Attribute

When you define a CMD job to run a Windows command, consider the following points when specifying the command attribute:

Permission level of the job owner:

When the command defined in the attribute affects a file system that supports Windows security mechanisms (for example, NTFS), the job's owner must have read and execute permission for that command. The job's owner must also have access to all resources and files referenced in the command.

Syntax restrictions:

- You cannot redirect standard input, output, and error files in the command attribute. Use the std_in_file, std_out_file, and std_err_file attributes instead.
- When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.
- Environment variables are not currently supported in the command attribute for Command jobs that run on Windows.

Redirecting the standard error file:

If the Windows command specified in the job definition does not exist, the job does not run. The standard error file is not created and the job log indicates that the error file is not found.

Starting a job interactively:

By default, all Command jobs start in batch mode. You can define a CMD job to start interactively on Windows. Interactive mode lets users view and interact with jobs that invoke Windows Terminal Services or user interface processes. To start a job interactively, specify the **interactive: y** attribute in your job definition.

Using the environment in the job profile:

- Environment variables for the command are defined by a default profile or the profile specified in the job definition.
- Although system environment variables are automatically set in the command's environment, user environment variables are not. You must define all other required environment variables in the job's profile.
- If a command works properly when issued at the command prompt, but fails to run properly when specified as a command attribute, the necessary user-defined environment variables and the variables defined in the job profile are probably different. If this is the case, use the Profiles Manager to verify that all required user environment variables are defined in the job profile.

Running Windows executable files:

- You can run the following executable file types using the command attribute:
 - Binary files (.exe)
 - Batch files (.bat, .cmd)
 - Command files (.com)
- To run Windows applications, such as ipconfig.exe, specify the command file in the command attribute, as shown in the following example:

```
command: "c:\winnt\system32\ipconfig.exe"
```
- To run Windows operating system commands, such as DIR or TYPE, ask your agent administrator to set the following parameters in the agentparm.txt file to true:

```
oscomponent.lookupcommand=true  
oscomponent.cmdprefix.force=true
```

When these parameters are set, the path to the command interpreter is automatically prefixed to the command. Therefore, you can specify the command and arguments in the command attribute (for example, command: "dir c:\temp\"). If these agent parameters are *not* set, you must explicitly invoke the command interpreter in the command attribute, as shown in the following example:

- ```
command: "c:\winnt\system32\cmd.exe /c dir c:\temp\"
```
- You can run other types of executable files from the Windows command line than those that are directly supported by the command attribute. (The executable files you can run are defined in the Windows system variable PATHEXT.) There are two ways to run those executable files using the agent:
    - Run the executable from a batch file (.bat) or command file (.cmd)
    - Invoke the program's interpreter and pass the command as a parameter using the command attribute.
  - For some GUI applications, if there is a command line interface, you can run the application by specifying the batch file name using the command attribute. For example, you can transfer files using the FTP DOS commands or send e-mail in a batch file using software such as BatMail.

Specifying a directory other than the agent's working directory:

By default, the agent's working directory is the directory where you install the agent. If your executable file inputs or outputs to a different directory, or runs another executable file in a different directory, specify that directory. For example, if your batch file runs an executable file, you can do the following:

- Specify the full path of the executable within the batch file:

```
C:\Program files\prog\espcmd1.bat
C:\Program files\prog\espcmd2.bat
C:\payroll\data\pay.exe
```

- Use the Change Directory (cd) command to switch to the directory:

```
cd C:\Program files\prog
espcmd1.bat
espcmd2.bat
cd C:\payroll\data
pay.exe
```

If your batch file takes input and output information, use the cd command to switch to the directory where the input and output files reside. In this example, file ftptest.bat reads the ftp command from the file ftpscript.src. The batch file outputs to file ftp.out in the current working directory.

```
cd d:\temp\script
ftp -n -i -s:ftpscript.src 131.50.30.28 >ftp.out
user guest
verbose
lcd d:\temp\data
cd /u1/guest
get data1
quit
```

If you run ftptest.bat from the command line, manually switch to the d:\temp\script directory before running the file. The Windows operating system knows the current working directory, so you do not need to include the cd d:\temp\script command in the batch file.

If you run ftptest.bat using the agent, however, the agent does not know the working directory. Specify the cd d:\temp\script command in the batch file, or the ftp command will fail.

Specifying the command or script name without the full path:

The command attribute usually requires the full path to the command or script name you want to run. However, you can specify the command or script name without the full path if all of the following conditions are true:

- The agent is configured to search for paths to command or script files.
- The script or command file is located in one of the following directories:
  - The directory the agent is installed in
  - WINDOWS\system32 directory on the agent computer
  - WINDOWS\system directory on the agent computer
  - WINDOWS directory on the agent computer
  - Any other directory whose path is set in the system path or user path on the agent computer

## Examples: Specifying the command Attribute on Windows

The following examples are Command Jobs on Windows:

### Example: Run a Command on Windows

This example runs the as\_test command on the Windows machine named winagent. The as\_test command waits for 5 seconds and exits with a return code of 0.

```
insert_job: win_cmd
job_type: CMD
machine: winagent
command: as_test -t 5 -e 0
```

### Example: Run a Batch File on Windows

This example runs the Backup batch file that is located in the C:\COMMON directory. The job runs on the Windows machine named winagent. If C:\COMMON is included in the run-time environment, either through the system environment variables or a job profile, you do not have to specify the full path to the file. Instead, you can specify the file name only (for example, command: Backup).

```
insert_job: win_batch
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
```

#### **Example: Remove All Files from a Subdirectory on Windows**

This example removes all files from the tmp subdirectory on the winagent machine. The tmp subdirectory is located in the directory specified in the MY\_BACKUPS global variable.

```
insert_job: win_remove
job_type: CMD
machine: winagent
command: c:\WINDOWS\system32\cmd.exe "/c del /s /q $$\{MY_BACKUPS\}\tmp*.*"
```

#### **Example: Run a Visual Basic Script on Windows**

In this example, the agent invokes the Visual Basic interpreter wscript.exe and passes the Visual Basic script notify.vbs as a parameter using the command attribute.

```
insert_job: win_vbjob
job_type: CMD
machine: winagent
command: "c:\winnt\system32\wscript.exe" "c:\programs\vbs\notify.vbs"
```

#### **Example: Run the Windows Notepad Application in Interactive Mode**

This example runs a Command job in interactive mode on Windows. The job opens the config.txt file in the Windows notepad application on the Windows desktop.

```
insert_job: edit_file
job_type: CMD
machine: winagent
description: "Edit/review a configuration file"
command: notepad.exe "c:\run_info\config.txt"
interactive: y
```

## condition Attribute—Define Starting Conditions for a Job

The condition attribute specifies job dependencies to use as starting conditions for the job you are defining. The job cannot run until all specified dependencies evaluate to true. You can base dependencies on one or more of the following:

- Job status; for example, success(DB\_BACKUP)
- Job status across instances; for example, success(jobB^PRD)
- Job exit code; for example, exitcode (my\_job)=4
- Jobs status with look-back; for example, success (job1,01.00)
- Jobs status across instances with look-back; for example, success (jobA^PRD,24)
- Jobs exit code with look-back; for example, exitcode (my\_job2,12)=4
- Jobs exit code across instances with look-back; for example, exitcode (my\_job3^PRD,02.30)=1
- Global variables; for example, VALUE(TODAY)=Friday

These conditions are described in the sections that follow.

### Supported Job types

This attribute is optional for all job types.

### Syntax

This attribute has the following formats:

condition: [()condition[]][(AND|OR)[()condition[]]]

condition: [()condition,look\_back[]][(AND|OR)[()condition,look\_back[]]]

#### **condition**

Defines any combination of dependencies based on job status, job exit codes, or global variables.

**Limits:** Up to 4096 characters

#### **Notes:**

- To configure complex conditions, combine a series of conditions with the AND or the OR logical operators. You can use the pipe symbol (|) instead of the word OR, and the ampersand symbol (&) instead of the word AND.
- Spaces between conditions and delimiters are optional.

- You can specify complex conditions by grouping the expressions in parentheses. The parentheses force precedence, and the equation is evaluated from left to right. For example, the following condition definition lets the dependent job start when either JobA and JobB return SUCCESS or JobD and JobE return SUCCESS, FAILURE, or TERMINATED:

condition: (success(JobA) AND success(JobB)) OR (done(JobD) AND done(JobE))

- In multiple conditions (that is, if JobC is dependent on SUCCESS of JobA and SUCCESS of JobB), JobC runs when both JobA and JobB succeed, no matter when the first SUCCESS event occurs. This may not be the appropriate result, for example, for a daily processing cycle where JobA finished yesterday but did not run today, and JobB succeeded today. To avoid this result, group the jobs in a box.

#### *look\_back*

Defines the last run of the condition or predecessor job. To support look-back in CA Workload Automation AE, you must add an additional operand to the condition statement.

##### **Notes:**

- The look-back feature is applied to job status, cross instance or external dependency job status, and exit code condition types only. Look-back is **not** applied to the global variable condition type.
- For more information about the look-back syntax and how to use it, see [Look-Back Dependencies](#) (see page 344).

##### **Notes:**

- If you specify a condition for an undefined job, the condition evaluates as false and any jobs dependent on the condition do not run. Use job\_depends to check for this type of invalid condition statement.
- You can use the autorep -J command to generate a report that lists the job definitions stored in the database. The autorep command inserts spaces between the job's conditions so that the report is easy to read. For example, if you specify **condition: s(a)&s(b)&s(c)** in your job definition, autorep outputs **condition: s(a) & s(b) & s(c)** in the report. If you want to re-insert a job using the definition shown in the autorep report, ensure that the total number of characters does not exceed 4096. Otherwise, the re-inserted job will fail.

## Job Status Dependencies

You can define a starting condition for a job so that it starts when another job returns a specific status. For example, you might specify that JobB starts when JobA returns a SUCCESS status and JobC starts when JobB returns a SUCCESS status.

To define a starting condition based on a job's status on CA Workload Automation AE, use the following format in the condition attribute:

`status(job_name)`

### **status**

Specifies one of the following:

#### **Done**

Indicates that the job you are defining may run when *job\_name*'s status is SUCCESS, FAILURE, or TERMINATED.

**Note:** You can specify d instead of Done.

**Example:** `d(jobA)`

#### **Failure**

Indicates that the job you are defining may run when *job\_name*'s status is FAILURE.

**Note:** You can specify f instead of Failure.

**Example:** `f(jobA)`

#### **Notrunning**

Indicates that the job you are defining may run when *job\_name*'s status is any value except RUNNING. Set the status to notrunning to prevent the dependent job from running at the same time as *job\_name*.

**Note:** You can specify n instead of Notrunning.

**Example:** `n(jobA)`

#### **Success**

Indicates that the job you are defining may run when *job\_name*'s status is SUCCESS.

**Note:** You can specify s instead of Success.

**Example:** `s(jobA)`

### Terminated

Indicates that the job you are defining may run when *job\_name*'s status is TERMINATED. A status of TERMINATED means the job was killed.

**Note:** You can specify t instead of Terminated.

**Example:** t(jobA)

### *job\_name*

Specifies the name of a job. The job you are defining depends on this job.

#### Notes:

- You can use the max\_exit\_success attribute to control the value of the SUCCESS status for a specific job. When you define the max\_exit\_success attribute, a job that exits with an exit code less than or equal to the specified value is treated as a success. FAILURE means the job exited with an exit code higher than the max\_exit\_success value. All other status settings are internally defined and you cannot control them.
- You can use uppercase or lowercase characters to define conditions. You cannot mix case.

**Windows Note:** Windows does not support the concept of process groups. When you issue a KILLJOB event for a job that runs an executable (\*.exe), KILLJOB kills the process specified in the command definition. When you issue a KILLJOB event for a job that runs something other than an \*.exe (for example, \*.bat, \*.cmd, or \*.com), KILLJOB terminates only the CMD.EXE process that CA Workload Automation AE used to launch the job. The Job Status is set according to the return code of the killed CMD.EXE process and can be one of the following: SUCCESS, FAILURE, or TERMINATED. Processes launched by user applications or batch (\*.bat) files are not killed.

#### Example: Define Starting Conditions Based on Job Success

This example (specified in the job definition for JobA) runs JobA if JobB succeeds.

condition: success(JobB)

#### Example: Define Starting Conditions Based on the Status of Multiple Jobs

This example (specified in the job definition for JobC) starts JobC only when both JobA and JobB complete successfully *or* when both JobD and JobE complete (regardless of whether they failed, succeeded, or terminated).

condition: (success(JobA) AND success(JobB)) OR (done(JobD) AND done(JobE))

**Example: Define Starting Conditions Based on Job Failure**

This example (specified in the job definition for JobA) runs JobA if JobB fails.

```
condition: failure(JobB)
```

**Example: Define Starting Conditions Based on Job Success in a Given Duration**

This example (specified in the job definition for JobA) runs JobA if JobB has completed successfully in the last 12 hours.

```
condition: success(JobB,12)
```

**Example: Define Starting Conditions Based on the Status of Multiple Jobs in a Given Duration**

This example (specified in the job definition of JobA) runs JobA if JobB has failed in the last 30 minutes and JobC has completed successfully on external instance XYZ in the last 90 minutes.

```
condition: failure(JobB,00.30) AND success(JobC^XYZ,01.30)
```

**Example: Define Starting Conditions with the notrunning Operator**

One use of the notrunning operator could be to avoid a database dump (DB\_DUMP) and a file backup (BACKUP) at the same time, which would cause frequent hard disk access. This example (specified in the job definition for JobA) runs JobA only when both of these resource-intensive jobs are **not** running.

```
condition: notrunning(DB_DUMP) AND notrunning(BACKUP)
```

## Cross-Instance Job Dependencies

To define a cross-instance job dependency, add the following attribute to your job definition:

**condition: *status(JOB\_NAME^INS)* [AND|OR *status(JOB\_NAME^INS)* ...]**

### **status**

Specifies the status that the external job must have before your job starts. When the external job completes with the specified status, the job dependency is satisfied. Options are the following:

- done
- failure
- notrunning
- success
- terminated

### ***JOB\_NAME***

Specifies the name of the external job that the local job depends on.

**Example:** JOBA

**CA AutoSys WA Connect Option Notes:** Due to naming limitations in the mainframe environment, the names of jobs specified as job dependencies between CA Workload Automation AE and CA AutoSys WA Connect Option must follow these guidelines:

- The first character of a job name must be an uppercase letter (A-Z), a pound sign (#), an at sign (@), or a dollar sign (\$).
- The remaining characters in the job name can be any combination of uppercase letters (A-Z), numbers (0-9), pound signs (#), at signs (@), and dollar signs (\$).
- All letters (A-Z) must be in uppercase.
- Job names can be up to 8 alphanumeric characters.

**CA UJMA Notes:** If the external job runs on a distributed system, the names of jobs specified as job dependencies between CA Workload Automation AE and CA UJMA must follow these guidelines:

- If the operating system permits, the job name can be up to 64 alphanumeric characters.
- If the operating system permits, the job name can be mixed case.
- The job name cannot contain blank spaces or tab characters.

If the external job runs on the mainframe, the names of jobs specified as job dependencies between CA Workload Automation AE and CA UJMA must follow the same guidelines as previously described for CA AutoSys WA Connect Option.

^

Indicates that the job resides on an external instance.

**INS**

Specifies the ID of the external instance that *JOB\_NAME* runs on.

**Limits:** The value must be three uppercase alphanumeric characters.

**Example:** PRD

**Note:** You can specify multiple job dependencies in a definition.

**Example: Specify a Cross-Instance Job Dependency**

This example defines a job that runs only when the following starting conditions are met:

- jobA on the same instance returns SUCCESS
- jobB on the external instance PRD returns SUCCESS

```
insert_job: dep_job1
machine: localhost
job_type: CMD
command: sleep 100
condition: success(jobA) AND success(jobB^PRD)
```

**Example: Define Starting Conditions Based on a Cross-Instance Job Status**

This example (specified in the job definition for JobA) runs JobA only when the job DB\_BACKUP (which resides on another CA Workload Automation AE instance named PRD) succeeds.

```
condition: success(DB_BACKUP^PRD)
```

## Exit Code Dependencies

You can base job dependencies on exit codes that indicate completed tasks. For example, you can specify that the system runs JobB if JobA fails with an exit code of 4.

To define a starting condition based on a job's exit code, use the following format in the condition attribute:

`exitcode (job_name) operator value`

***job\_name***

Specifies the name of a job. The job you are defining depends on this job.

**Limits:** Up to 64 characters

***operator***

Specifies one of the following comparison operators:

= Equals

!= Does not equal

< Less than

> Greater than

<= Less than or equal to

>= Greater than or equal to

***value***

Defines the value to compare against.

**Limits:** Up to 14 characters

### Example: Define Starting Conditions Based on Job Exit Code

This example (specified in the job definition for JobB) runs JobB when JobA completes with an exit code of 4.

`condition: exitcode (JobA) = 4`

## Global Variable Dependencies

You can base job dependencies on global variables set using the sendevent command. When using global variables, the value of the variable must evaluate to TRUE to satisfy the job dependency.

To define a starting condition based on a global variable, use the following format in the condition attribute:

`VALUE(global_name) operator value`

### *global\_name*

Identifies a global variable that is already defined.

**Limits:** Up to 30 alphanumeric characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-)

### *operator*

Specifies one of the following comparison operators:

= Equals

!= Does not equal

< Less than

> Greater than

<= Less than or equal to

>= Greater than or equal to

### *value*

Defines the value to compare against.

**Limits:** Up to 100 alphanumeric characters. Values with embedded blanks must be enclosed with quotation marks.

**Example:** `value(VAR1)="SOME DATA"`

### Example: Define Starting Conditions Based on a Global Variable Value

This example (specified in the job definition for JobA) runs JobA only if the value of the global variable OK\_TO\_RUN is greater than 2.

`condition: VALUE(OK_TO_RUN)>2`

### Example: Define Starting Conditions Based on Multiple Condition Types

This example (specified in the job definition for JobA) runs JobA only if the job BACKUP completes with a SUCCESS status and the global variable TODAY has a value of Friday.

`condition: success(BACKUP) AND value(TODAY)=Friday`

## Look-Back Dependencies

The look-back feature is defined as the last run of the condition or predecessor job. The last run is defined as the ending time of the last successful run of the job. If the condition or predecessor job has run successfully after the job which has the condition or predecessor job defined, then the condition or predecessor is satisfied and the job starts. If not, then the condition or predecessor job is not satisfied and the job for which the predecessor job condition is defined is not started.

To support the look-back feature, you must add an additional operand to the condition statement. You can apply the look-back feature to job status, cross-instance or external dependency job status, and exit code condition types only. You cannot apply it to the global variable condition type.

To define a starting condition based on a look-back value, use the following format in the condition attribute:

```
status(job_name, hhhh.mm)
status(job_name^INS, hhhh.mm)
exitcode(job_name, hhhh.mm) operator value
exitcode(job_name^INS, hhhh.mm) operator value
hhhh
```

Indicates the hours taken for the last run of the condition or predecessor job. You can look back approximately 416.58 days.

**Limits:** 0-9999

**mm**

Indicates the minutes taken for the last run of the condition or predecessor job.

**Limits:** 0-59 when specifying minutes; 0-9998 for *hhhh* when specifying hours and minutes (for example, 9998.59)

**Examples:** 0, 00.15, 06.30, 23.30, 24, 98.30, 720, 9998.59, 9999

### Notes:

- When the elapsed time to be specified is less than one hour, you must specify the hourly value (*hhhh*) as "00". For example, "00.30" represents 30 minutes. A value of ".30" indicates an elapsed time of 30 hours. Specifying an elapsed time of ".30" is not valid.
- When you specify 0, CA Workload Automation AE examines the last end time of the job first. It then examines the last end time of the condition job. If the condition job has run since the last run of the job for which the condition is coded for, the job is allowed to start. If the condition job has not run since the last run of the job for which the condition is coded for, the job is not allowed to start.

- When you specify **9999**, CA Workload Automation AE looks back indefinitely (the default behavior of Unicenter AutoSys JM r4.5 and previous releases). If the condition job has run, the job is allowed to start. This is the same as not specifying a look-back value.
- Instead of using a period (.) to separate *hhhh* and *mm*, you can use a colon that must be escaped (\:) as shown in the following example:  
`condition: success(JobA,01\:00) and failure(JobB,02\:15)`  
You must use an escape character (backslash) with any colons used in an attribute's value because JIL parses on the combination of keywords followed by a colon.
- The autorep command outputs the look-back value when producing a detailed report using the "-q" option.

#### Example: Specify a Look-back Dependency

This example starts the dependent job if JobA is successful during the last hour and JobB failed during the last 2 hours and 15 minutes.

```
condition: success(JobA,01.00) and failure(JobB,02.15)
```

## condition\_code Attribute—Specify a Code to Indicate Success or Failure

The condition\_code attribute specifies a condition code that indicates success or failure. You can define condition codes for jobs, steps, procedure steps, or programs.

For each condition code or range of condition codes, you can indicate whether the scheduling manager interprets the code or range as a success or a failure. You can also indicate whether the job continues or stops running. Each specification becomes a separate item. Together these specifications form a list of condition codes.

### Supported Job Type

This attribute is optional for the [z/OS Regular \(ZOS\) job type](#) (see page 285).

### Syntax

This attribute has the following format:

```
condition_code: rc=return_code
 [,result=SUCCESS|FAILURE]
 [,action=CONTINUE|STOP]
 [,proc_step=procedure_name]
 [,program=program_name]
 [,step_name=step_name]
```

#### **rc=return\_code**

Specifies the return code using one of the following values:

- A condition code for one number or a range of numbers between 1 and 4095 (ranges are separated by colons, such as 1:4095).
- A system abend code (Sccc), such as SOC1 or SB37.
- A user abend code (Unnnn), such as U0001 or U0462. The nnnn must be four decimal digits and cannot exceed 4095.

**Limits:** Up to 9 characters

#### **result=SUCCESS|FAILURE**

(Optional) Specifies whether to consider the job as failed based on the condition code. Options are the following:

##### **SUCCESS**

Considers the job not failed if the condition code is matched. This is the default.

##### **FAILURE**

Considers the job failed if the condition code is matched.

**action=CONTINUE|STOP**

(Optional) Specifies whether the job continues with the next step regardless of whether the job is considered as failed. Options are the following:

**CONTINUE**

Continues with the next step if the condition code is matched. This is the default.

**STOP**

Stops the job if the condition code is matched.

**proc\_step=procedure\_name**

(Optional) Specifies a valid z/OS procedure name.

**Limits:** Up to eight characters; first character must not be numeric

**program=program\_name**

(Optional) Specifies a valid z/OS program name.

**Limits:** Up to eight characters; first character must not be numeric

**step\_name=step\_name**

(Optional) Specifies a valid z/OS step name.

**Limits:** Up to eight characters; first character must not be numeric

**Notes:**

- You can specify multiple condition codes. Use a separate condition\_code attribute for each code. For example, to specify the 111 and 112 condition codes for your job, define a condition\_code attribute for 111 and a condition\_code attribute for 112.
- The entire value can be up to 4096 characters.

#### **Example: Specify Multiple Condition Codes**

This example runs a z/OS Regular job. The job is considered failed if the return code is 114 or 115. The job is not considered failed if the return code is 111, 112, or 113. The job continues with the next step only if the return code is 111, 112, or 114. Otherwise, the job stops running.

```
insert_job: zos_job
job_type: ZOS
machine: zagent
jcl_library: CYBJAK1.JCL
jcl_member: CYBJAK11
condition_code: rc=111,result=SUCCESS,action=CONTINUE,step_name=xyz
condition_code: rc=112,result=SUCCESS,action=CONTINUE
condition_code: rc=113,result=SUCCESS,action=STOP
condition_code: rc=114,result=FAILURE,action=CONTINUE
condition_code: rc=115,result=FAILURE,action=STOP
```

## connect\_string Attribute—Specify Database Resource Location

The SQL attribute specifies the database resource location for a database job.

### Supported Job Types

This attribute is optional for the following job types:

- [Database Monitor \(DBMON\)](#) (see page 207)
- [Database Stored Procedure \(DBPROC\)](#) (see page 209)
- [Database Trigger \(DBTRIG\)](#) (see page 211)
- [SQL](#) (see page 272)

### Syntax

This attribute has the following format:

```
connect_string: "dburl"
"dburl"
```

Specifies a JDBC database resource location for a job. Use the appropriate format for your database type:

- For an Oracle database:

```
connect_string: "jdbc:oracle:thin:@host:port:dbname"
```

- For a Microsoft SQL Server database:

```
connect_string: "jdbc:sqlserver://host:port;DatabaseName=dbname"
```

- For an IBM DB2 database:

```
connect_string: "jdbc:db2://host:port/dbname"
```

- For a Sybase database:

```
connect_string: "jdbc:sybase:Tds:host:port/dbname"
```

**Limits:** Up to 256 characters; case-sensitive

### Notes:

- If you do not specify this attribute in the job definition, a default database resource location must be defined in the db.default.url parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- The connect\_string attribute overrides the default resource location specified in the agent's agentparm.txt file.

### **Example: Specify an Oracle Database**

This example uses the Oracle database named ORDERS that is reachable on port 1433 on host 172.31.255.255.

```
insert_job: QRY1
job_type: SQL
machine: DB_agent
owner: sys@orcl
sql_command: SELECT * from NEWORDS
destination_file: "c:\log\qry1.txt"
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

### **Example: Specify a Microsoft SQL Server Database**

This example uses the Microsoft SQL Server database named ORDERS that is reachable on port 1433 on a host named myhost.

```
insert_job: QRY2
job_type: SQL
machine: DB_agent
owner: sa@myhost
sql_command: SELECT * from NEWORDS
destination_file: "c:\log\qry2.txt"
connect_string: "jdbc:sqlserver://myhost:1433;DatabaseName=ORDERS"
```

### **Example: Specify an IBM DB2 Database**

This example updates a record in the STAFF table under the user entadm. The job changes the years to 3 for the employee with the name Jonson.

```
insert_job: UPDATE
job_type: SQL
machine: DB_AGENT
owner: entadm@myhost
sql_command: UPDATE ENTADM.STAFF SET YEARS=3 where NAME='Jonson'
destination_file: "c:\log\updatejob.txt"
connect_string: "jdbc:db2://172.31.255.255:50000/SAMPLE"
```

#### Example: Specify a Sybase Database

This example monitors the ap\_invoices table for changes. The job connects to the Sybase database named APDB that is reachable on port 5001 on a host named myhost. The job completes when the table has changed.

```
insert_job: db_sqltrig_upd
job_type: DBTRIG
machine: localhost
dbtype: Sybase
connect_string: "jdbc:sybase:Tds:myhost:5001/APDB"
trigger_type: UPDATE
tablename: ap_invoices
owner: admin@myhost
```

## connection\_factory Attribute—Specify the JNDI Name of the Connection Factory

The connection\_factory attribute specifies the connection factory JNDI name in a JMS job.

#### Supported Job Types

This attribute is required for the following job types:

- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)

#### Syntax

This attribute has the following format:

connection\_factory: *factory*

#### *factory*

Specifies the connection factory JNDI name. The connection factory contains all the bindings needed to look up the referenced topic or queue. JMS jobs use the connection factory to create a connection with the JMS provider.

**Limits:** Up to 256 characters; case-sensitive

### **Example: Specify Connection Factory in a JMS Publish Job**

This example publishes the message "this is my message" to the queue named Queue. The Java class of the message is String. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere MQ server and 2809 is the ORB port. The job uses the cyberuser JNDI user name to gain access to the connection factory named ConnectionFactory.

```
insert_job: publish
job_type: JMSPUB
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
connection_factory: ConnectionFactory
destination_name: Queue
use_topic: FALSE
message_class: String
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="this is my message"
```

**Note:** The agent does not support JMS messaging on IBM WebSphere. If you have IBM WebSphere MQ, your agent administrator can set up the agent plug-in to run JMS Publish and JMS Subscribe for JMS queues. JMS topics are not supported on IBM WebSphere MQ.

## continuous Attribute—Specify Whether the Job Monitors Continuously

The continuous attribute specifies whether a job continuously monitors for a condition.

**Note:** If you do not specify the continuous attribute in your job definition, the job completes the first time the condition is met (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [Database Monitor \(DBMON\)](#) (see page 207)
- [Database Trigger \(DBTRIG\)](#) (see page 211)
- [File Trigger \(FT\)](#) (see page 217)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [JMX-MBean Subscribe \(JMXSUB\)](#) (see page 240)
- [SAP Event Monitor \(SAPEVT\)](#) (see page 261)
- [SAP Process Monitor \(SAPPMM\)](#) (see page 264)

**Note:** This attribute does not apply to File Trigger jobs if EXIST or NOTEXIST is specified for the watch\_file\_type attribute.

### Syntax

This attribute has the following format:

continuous: Y | N

**Y**

Monitors for the conditions continuously. Each time the specified conditions occur, an alert is written to the scheduler log file (event\_demon.\$AUTOSERV on UNIX and event\_demon.%AUTOSERV% on Windows).

#### Notes:

- To end continuous monitoring, you must complete the job manually by issuing the following command:  
`sendevent -E KILLJOB -J job_name`
- To specify continuous: Y for SAP Event Monitor jobs, you must specify N in the sap\_is\_trigger attribute.

**N**

Specifies that the job completes the first time the condition is met. This is the default.

**File Trigger Notes:**

- If the file trigger type is CREATE, EXPAND, SHRINK, or UPDATE, the File Trigger job will not complete unless it is forced to complete.
- If the file trigger type is DELETE, the File Trigger job completes when all the monitored files are deleted or it is forced to complete.
- The following applies to using wildcards in the file name value with the continuous attribute:
  - If the file trigger type is CREATE or DELETE, the file name value must contain a wildcard (\* or ?), for example, /usr/data/pay\*.
  - If the file trigger type is EXPAND or UPDATE and the file name value contains a wildcard, the agent monitors all files that match the file name criteria when the file trigger is set. When the criteria is satisfied for any of the selected files, the job triggers and the agent continues to monitor the files.

**Note:** If the continuous attribute is specified with the watch\_no\_change attribute and a wildcard is used in the file name value, all of the matching files must remain unchanged for the duration of the watch\_no\_change interval. If another file satisfying the trigger criteria is updated during the watch\_no\_change interval, the trigger waits until the trigger interval lapses on the first file before monitoring the second file for the duration of the watch\_no\_change interval.

**Example: Continuously Monitor the Same File Using Multiple File Trigger Jobs**

In this example, two File Trigger jobs monitor the same file for a change in size by 10 KB. One job monitors for an increased change in size and the other job monitors for a decreased change in size. The jobs are independent and do not relate to each other in any way.

```
insert_job: ftjob1
job_type: FT
machine: ftagent
watch_file: "c:\data\totals"
watch_file_type: EXPAND
continuous: Y
watch_file_change_type: DELTA
watch_file_change_value: 10240 /* The value must be entered in bytes (10 x 1024 bytes
= 10240) */
```

```
insert_job: ftjob2
job_type: FT
machine: ftagent
watch_file: "c:\data\totals"
watch_file_type: SHRINK
continuous: Y
watch_file_change_type: DELTA
watch_file_change_value: 10240 /* The value must be entered in bytes (10 x 1024 bytes
= 10240) */
```

Suppose that the initial file size of totals (c:\data\totals) is 100 KB and it changes as follows:

100 KB (initial size), 80 KB, 90 KB, 110 KB, 50 KB, 60 KB

The following triggers occur:

- Because the file trigger type is EXPAND, the first job (ftjob1) writes an alert to the scheduler log file once when the file size changes from 90 KB to 110 KB.
- Because the file trigger type is SHRINK, the second job (ftjob2) writes an alert to the scheduler log file twice. The triggers occur when the file shrinks from 100 KB to 80 KB and then again when the file shrinks from 110 KB to 50 KB.

The jobs end when they are forced to complete.

## copy\_jcl Attribute—Specify a Data Set Name that Receives the Copy JCL

The copy\_jcl attribute specifies the data set name that receives the working copy of the JCL you submitted.

### Supported Job Type

This attribute is optional for the [z/OS Regular \(ZOS\) job type](#) (see page 285).

### Syntax

This attribute has the following format:

`copy_jcl: data_set_name`

#### *data\_set\_name*

Specifies the data set name that receives the working copy of the JCL you submitted. Data set names typically have up to four positions; positions must each be eight characters or less and separated by periods.

**Limits:** Up to 44 characters

**Example:** prod.copyjcl.weekly

### Example: Store a Working Copy of the JCL that You Submitted

Suppose that the agent ZOS1 submits the JCL in member CYBDL01A in the CYBDL01.JCLLIB library. If the job fails, you can modify a working copy of the JCL in the CYBDL01.COPY.JCLLIB data set, and resubmit the job without affecting the JCL source.

```
insert_job: CYBDL01A
job_type: ZOS
machine: ZOS1
jcl_library: CYBDL01.JCLLIB
jcl_member: CYBDL01A
owner: CYBDL01
copy_jcl: CYBDL01.COPY.JCLLIB
```

## create\_method Attribute—Specify a Create Method

The `create_method` attribute specifies the name of the create method in a Session Bean job.

### Supported Job Type

When accessing a stateful session bean, this attribute is required for the [Session Bean \(SESSBEAN\) job type](#) (see page 270).

### Syntax

This attribute has the following format:

`create_method: create_method`

#### *create\_method*

Specifies the name of the create method.

**Default:** `create`

**Limits:** Up to 256 characters; case-sensitive; it must always begin with `create`

**Example:** `createaccount`

### Example: Create a Stateful Session Bean for an Online Shopping Cart

Suppose that you want to access a stateful session bean for an online shopping cart. The `createaddbook` method creates the `Shoppingcart` stateful bean for the duration of the job. The `addbook` method adds books to the shopping cart using the book's ISBN number. In this example, the Session Bean job adds two books to the shopping cart with ISBN numbers 1551929120 and 1582701709. When the job runs, two books are added to the shopping cart.

```
insert_job: addbook
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: ShoppingCart
create_method: createaddbook
method_name: addbook
create_parameter: String="ISBN"
j2ee_parameter: Integer[2]="1551929120,1582701709"
```

## create\_name Attribute—Specify a Create Method

The `create_name` attribute specifies the name of the create method in an Entity Bean job.

**Note:** If you do not specify the `create_name` attribute in your job definition, the job uses `create` for the create method (the default).

### Supported Job Type

When the operation type is CREATE, this attribute is optional for the [Entity Bean \(ENTYBEAN\) job type](#) (see page 214).

### Syntax

This attribute has the following format:

`create_name: create_method`

#### `create_method`

Specifies the name of the create method.

**Default:** `create`

**Limits:** Up to 256 characters; case-sensitive; it must always begin with `create`

**Example:** `createaccount`

### Example: Create an Entity Bean

Suppose that you want to create an entity bean that stores information about a customer such as the customer ID and phone number. The initial context factory supplied by the JNDI service provider is `weblogic.jndi.WLInitialContextFactory`. The service provider's URL is `t3://localhost:7001`, where `localhost` is the domain name of the WebLogic application server and `7001` is the ORB port. When the job runs, the entity bean instance is created.

```
insert_job: create
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
create_name: createcustomer
operation_type: CREATE
create_parameter: String="customerid", String="800-555-0100"
```

## create\_parameter Attribute—Specify Create Parameters

The create\_parameter attribute specifies the create parameters in an Entity Bean and Session Bean job. You can specify create parameters to create an entity bean in a relational database on your application server or to access a stateful session bean.

**Note:** To access stateful session beans, you must specify create parameters.

### Supported Job Type

This attribute is optional for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

### Syntax

This attribute has the following format:

```
create_parameter: type=value | payload_job=job_name[, type=value |
payload_job=job_name...]
```

#### *type=value*

Specifies the Java class and String value for the parameter.

**Limits:** Up to 1024 characters; case-sensitive

**Examples:** String="5:00PM", String[2]="winnt1,winnt2", double=2000

**Note:** If the package of the Java class is not specified, java.lang is assumed.

#### *payload\_job=job\_name*

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

### Notes:

- You can define up to 64 create parameters using one create\_parameter attribute or multiple entries of create\_parameter.
- In each create\_parameter entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. Specify the parameters in the same order as they appear in the create method. The job fails if the parameters are listed in the wrong order.

#### **Example: Specify Create Parameter to Access a Stateful Session Bean**

Suppose that you want to access a stateful session bean for an online shopping cart. The createaddbook method creates the ShoppingCart stateful bean for the duration of the job. The addbook method adds books to the shopping cart using the book's ISBN number. In this example, the Session Bean job adds two books to the shopping cart with ISBN numbers 1551929120 and 1582701709. When the job runs, two books are added to the shopping cart.

```
insert_job: addbook
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: ShoppingCart
create_method: createaddbook
method_name: addbook
create_parameter: String="ISBN"
j2ee_parameter: Integer[2]="1551929120,1582701709"
```

## **cpu\_usage Attribute—Specify Whether to Monitor Available or Used CPU**

The `cpu_usage` attribute specifies whether the job monitors for available or used CPU.

**Note:** If you do not specify the `cpu_usage` attribute in your job definition, the job monitors for available CPU (the default).

#### **Supported Job Type**

This attribute is optional for the [CPU Monitoring \(OMCPU\) job type](#) (see page 206).

To use this attribute, you must specify WAIT or CONTINUOUS for the `monitor_mode` attribute. This attribute does not apply when the monitor mode is NOW.

#### **Syntax**

This attribute has the following format:

`cpu_usage: FREE | USED`

#### **FREE**

Specifies that the job monitors the available CPU processing capacity. This is the default.

#### **USED**

Specifies that the job monitors the used CPU processing capacity.

**Notes:**

- The CPU usage reported as greater than 100 percent occurs when a partitioned or virtualized machine has overcommitted its CPU allocation and was able to use unused CPU from the rest of the machine.
- On UNIX, the CPU calculations are based on load averaging as displayed by the command top. You can scale the results the agent returns with the objmon.cpu.scalefactor parameter. For more information about the objmon.cpu.scalefactor parameter, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- You must specify at least one of the operands FROM or TO in the CPU statement when using the WAITMODE statement set to the WAIT value.
- The CPU statement does not apply when using the WAITMODE statement set to the NOW value.

**Example: Monitor Used CPU**

This example monitors the used CPU on the unixagent computer. The job completes when the used CPU is less than 20 percent or greater than 80 percent.

```
insert_job: omcpu_job
job_type: OMCPU
machine: unixagent
lower_boundary: 20
upper_boundary: 80
inside_range: FALSE
cpu_usage: USED
```

## date\_conditions Attribute—Base Job Start on Date and Time Attribute Values

The date\_conditions attribute specifies whether to use the date or time conditions defined in the following attributes to determine when to run the job:

- [autocal\\_asc](#) (see page 55)
- [days\\_of\\_week](#) (see page 363)
- [exclude\\_calendar](#) (see page 380)
- [must\\_complete\\_times](#) (see page 493)
- [must\\_start\\_times](#) (see page 499)
- [run\\_calendar](#) (see page 606)
- [run\\_window](#) (see page 607)
- [start\\_mins](#) (see page 676)
- [start\\_times](#) (see page 677)
- [timezone](#) (see page 722)

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

date\_conditions: y | n

**y**

Uses the date and time conditions defined in other attributes to determine when to run the job.

**Note:** You can specify 1 instead of y.

**n**

Ignores the date and time conditions defined in other attributes to determine when to run the job. This is the default.

**Note:** You can specify 0 instead of n.

**Example: Base Job Start on Date and Time Attribute Values**

This example uses starting date and time conditions specified in the autocal\_asc, days\_of\_week, exclude\_calendar, run\_calendar, run\_window, start\_mins, start\_times, must\_start\_times, must\_complete\_times, and timezone attributes in the job definition.

```
date_conditions: y
```

## days\_of\_week Attribute—Specify which Days of the Week to Run a Job

The days\_of\_week attribute specifies one or more days of the week on which to run the job you are defining.

When you define the days\_of\_week attribute, you must also define either the start\_times or start\_mins attribute. CA Workload Automation AE schedules the job to run at the times specified in the start\_times or start\_mins attribute on the specified days of the week.

**Note:** The days\_of\_week attribute has no effect when you also define the run\_calendar attribute. The date\_conditions attribute controls the usage of this attribute.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following formats:

```
days_of_week: day [,day...]
```

```
days_of_week: all
```

***day***

Specifies the days of the week when the job runs. Options are the following:

- mo—Specifies Monday.
- tue—Specifies Tuesday.
- we—Specifies Wednesday.
- th—Specifies Thursday.
- fr—Specifies Friday.
- sa—Specifies Saturday.
- su—Specifies Sunday.

**Default:** No days are selected.

**Notes:**

- You can specify multiple days. Separate each day with a comma (,).
- The entire value can be up to 80 characters.

***all***

Specifies that the job runs every day of the week.

**[Example: Specify to Run a Job Only on Weekdays](#)**

This example specifies that the job runs only on weekdays.

```
date_conditions: y
days_of_week: mo, tu, we, th, fr
```

**[Example: Specify to Run a Job on Every Day of the Week](#)**

This example specifies that the job runs every day of the week.

```
date_conditions: y
days_of_week: all
```

## dbtype Attribute—Specify the Type of Database

The dbtype attribute specifies the type of the database that the job monitors. Depending on the database type, CA Workload Automation AE generates the proper syntax for the database trigger if it monitors multiple actions. For example, Microsoft SQL Server requires the actions to be separated by commas, and Oracle requires that the actions be separated by the "or" operand. CA Workload Automation AE automatically changes the actions to the proper syntax according to the dbtype value, if needed.

**Note:** This value is *not* used to validate the connected database is of the specified type.

### Supported Job Type

This attribute is required for the [Database Trigger \(DBTRIG\) job type](#) (see page 211).

### Syntax

This attribute has the following format:

dbtype: Oracle | MSSQL | DB2 | Sybase

#### Oracle

Specifies that the job monitors an Oracle database.

#### MSSQL

Specifies that the job monitors a Microsoft SQL Server database.

#### DB2

Specifies that the job monitors an IBM DB2 database.

#### Sybase

Specifies that the job monitors a Sybase database.

#### **Example: Specify the Oracle Database Type**

This example monitors the Inventory\_List table in the Oracle database named ORDERS. Because the dbtype is Oracle, CA Workload Automation AE changes the "INSERT,DELETE" syntax to "INSERT OR DELETE". When a row is inserted or deleted, the job completes.

```
insert_job: dbtrig_oracle
job_type: DBTRIG
machine: dbagent
tablename: Inventory_List
dbtype: Oracle
trigger_type: INSERT,DELETE
owner: sys@orcl
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

#### **Example: Specify the Sybase Database Type**

This example monitors the ap\_invoices table for changes. The job connects to the Sybase database named APDB that is reachable on port 5001 on a host named myhost. The job completes when the table has changed.

```
insert_job: db_sqltrig_upd
job_type: DBTRIG
machine: localhost
dbtype: Sybase
connect_string: "jdbc:sybase:Tds:myhost:5001/APDB"
trigger_type: UPDATE
tablename: ap_invoices
owner: admin@myhost
```

## **description Attribute—Define a Text Description for a Job**

The description attribute defines a text description for the job you are defining.

#### **Supported Job Types**

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

```
description: "text"
"text"
```

Defines an optional free-form text description of the job.

**Limits:** Up to 500 alphanumeric characters (including spaces)

#### Example: Define a Text Description for a Job

This example specifies that the job you are defining is an incremental daily backup of the database.

```
description: "incremental daily backup of the database"
```

## destination\_file Attribute—Specify an Output Destination File

The destination\_file attribute specifies the output destination file in a job. In a payload producing job, the file stores the Java serialized object produced by the job. In an SQL job, the file stores the SQL query results.

**Note:** If you omit this attribute in a payload producing job, the Java serialized object is stored in the spool directory on the agent computer using the job name and a numeric suffix as the file name. If you omit this attribute in an SQL job, the output is stored in the job's spool file.

### Supported Job Types

This attribute is optional for the following job types:

- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [POJO](#) (see page 252)
- [RMI \(JAVARMI\)](#) (see page 254)
- [Session Bean \(SESSBEAN\)](#) (see page 270)
- [SQL](#) (see page 272)
- [Web Service \(WBSVC\)](#) (see page 276)

### Syntax

This attribute has the following format:

```
destination_file: destination
```

***destination***

Specifies the output destination file. In a payload producing job, the file stores the Java serialized object produced by a job. In an SQL job, the file stores the SQL query results.

**Limits:** Up to 256 characters; case-sensitive

**Examples:** /tmp/myoutput, "D:\Stock output\STOCK3"

**Notes:**

- Your agent administrator can specify a default destination file for all jobs by setting the spooldir parameter in the agent's agentparm.txt file.
- The destination\_file attribute overrides the default file specified in the agent's agentparm.txt.

**Example: Save Output from a Session Bean Job in a File**

Suppose that you want to invoke the reverse method on the CybEJBTestBean stateless session bean. The reverse method has one parameter, with type java.lang.String and value a23. The output from the reverse method is saved in the C:\Makapt15 file. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere application server and 2809 is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

**Example: Return Data from a Database Table that Match a Condition**

This example queries a database table named emp for an emp\_name value that starts with S, followed by at least two characters. If the result starts with S, followed by one other character, the job completes. Otherwise, the job fails. The job's output is written to c:\temp\sql\_job1\_out.log.

```
insert_job: sql_job1
job_type: SQL
machine: SQL_server1
sql_command: select emp_name from emp where emp_name like 'S%'
connect_string:"jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
success_criteria: emp_name=S.*
destination_file: "c:\temp\sql_job1_out.log"
```

## destination\_name Attribute—Specify the JNDI Name of the Topic or Queue

The destination\_name attribute specifies the JNDI name of the topic or queue in a JMS job.

**Note:** Topics are not supported on IBM WebSphere or IBM WebSphere MQ.

### Supported Job Types

This attribute is required for the following job types:

- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)

### Syntax

This attribute has the following format:

`destination_name: destination`

#### *destination*

Specifies the JNDI name of the topic or queue. The job uses the JNDI name to indicate the destination where messages are received.

**Limits:** Up to 256 characters; case-sensitive

#### **Example: Monitor a Queue on a WebLogic Application Server**

This example continuously monitors the queue named Queue (residing on WebLogic) for a message matching the filter criteria. The consumed messages from the queue are stored in the file /export/home/user1/outputfile1. The service provider's URL is t3://172.24.0.0:7001, where 172.24.0.0 is the IP address of the WebLogic Application server and 7001 is the ORB port.

```
insert_job: monitor
job_type: JMSSUB
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://172.24.0.0:7001"
connection_factory: ConnectionFactory
destination_name: Queue
continuous: Y
filter: abc\s... \s[a-zA-Z]+\sFilter! [\sa-z0-9]+
use_topic: FALSE
destination_file: /export/home/user1/outputfile1
j2ee_user: cyberuser
```

In this example, the regular expression used as the filter criteria can be defined as follows:

abc\s... \s[a-zA-Z]+\sFilter! [\sa-z0-9]+

**abc\s**

Specifies the text abc, followed by white space.

**... \s**

Specifies any three characters, followed by white space.

**[a-zA-Z]+\s**

Specifies at least one letter, followed by white space.

**Filter![\sa-z0-9]+**

Specifies the text Filter!, followed by at least one of the following: white space or digit or lower case letter.

**Example:** abc vvv B Filter! 95

## disk\_drive Attribute—Specify the Disk Space to Monitor

The disk\_drive attribute specifies the disk space to monitor.

### Supported Job Type

This attribute is required for the [Disk Monitoring \(OMD\) job type](#) (see page 213).

### Syntax

This attribute has the following format:

disk\_drive: *drive*  
***drive***

Specifies the path to the disk, logical partition, or auxiliary storage pool to be monitored.

**Limits:** Up to 16 characters; case-sensitive

**Example:** /dev/QASP02

**UNIX/Windows:** Specify the path to the disk or logical partition to monitor.

**i5/OS:**

- Specify the path to a file system mounted on the i5/OS operating system.
- To specify the system ASP, type the forward slash (/).
- To specify any other ASP, use the following format, where *digit1* and *digit2* indicate the ASP number:

/dev/QASP*digit1**digit2*

### Example: Monitor Disk Space on UNIX

This example monitors /export/home for available space. The job completes when the available space is between 8 and 10 gigabytes (GB), and the available space changes by more than 15 percent. If the change in value is less than or equal to 15, the job does not register a change.

```
insert_job: omd_unix
job_type: OMD
machine: unixagt
disk_drive: /export/home
disk_space: FREE
disk_format: GB
lower_boundary: 8
upper_boundary: 10
inside_range: TRUE
no_change: 15
```

#### **Example: Monitor Disk Space on Windows**

This example monitors the C drive on a Windows computer for used space. When the value of disk space used falls below 16 percent or exceeds 95 percent, the job completes.

```
insert_job: omd_win
job_type: OMD
machine: winagent
disk_drive: C
disk_format: PERCENT
lower_boundary: 16
upper_boundary: 95
inside_range: FALSE
disk_space: USED
```

#### **Example: Monitor Disk Space on i5/OS**

This example monitors the system auxiliary storage pool on an i5/OS computer for used space. When 90 to 100 percent of the storage space is used, the job completes.

```
insert_job: omd_i5
job_type: OMD
machine:i5agent
disk_drive: /
disk_format: PERCENT
lower_boundary: 90
upper_boundary: 100
inside_range: TRUE
disk_space: USED
```

## **disk\_format Attribute—Specify the Unit of Measurement Used to Monitor Disk Space**

The disk\_format attribute specifies the unit of measurement used to monitor available or used disk space.

**Note:** If you do not specify the disk\_format attribute in your job definition, the job monitors for disk space in percent (the default).

#### **Supported Job Type**

This attribute is optional for the [Disk Monitoring \(OMD\) job type](#) (see page 213).

## Syntax

This attribute has the following format:

`disk_format: PERCENT | GB | MB | KB | B`

### PERCENT

Monitors disk usage by percentage. This is the default.

**Note:** If you specify PERCENT, the values specified by the lower\_boundary, upper\_boundary, and no\_change attributes can be greater than 100.

### **GB**

Monitors disk usage in gigabytes.

### **MB**

Monitors disk usage in megabytes.

### **KB**

Monitors disk usage in kilobytes.

### **B**

Monitors disk usage in bytes.

## [Example: Monitor Disk Space on UNIX](#)

This example monitors /export/home for available space. The job completes when the available space is between 8 and 10 gigabytes (GB), and the available space changes by more than 15 percent. If the change in value is less than or equal to 15, the job does not register a change.

```
insert_job: omd_unix
job_type: OMD
machine: unixagt
disk_drive: /export/home
disk_space: FREE
disk_format: GB
lower_boundary: 8
upper_boundary: 10
inside_range: TRUE
no_change: 15
```

## disk\_space Attribute—Specify Whether to Monitor Available or Used Disk Space

The disk\_space attribute specifies whether the job monitors for available or used disk space.

**Note:** If you do not specify the disk\_space attribute in your job definition, the job monitors for available disk space (the default).

### Supported Job Type

This attribute is optional for the [Disk Monitoring \(OMD\) job type](#) (see page 213).

### Syntax

This attribute has the following format:

disk\_space: FREE | USED

#### FREE

Specifies that the job reads the available disk space for monitoring.

#### USED

Specifies that the job reads the used disk space for monitoring.

**Note:** For a newly created file system on any platform, the operating system usually reports the file system overhead as used bytes.

### Example: Monitor Available Disk Space on UNIX

This example monitors /export/home for available space. The job completes when the available space is between 8 and 10 gigabytes (GB), and the available space changes by more than 15 percent. If the change in value is less than or equal to 15, the job does not register a change.

```
insert_job: omd_unix
job_type: OMD
machine: unixagt
disk_drive: /export/home
disk_space: FREE
disk_format: GB
lower_boundary: 8
upper_boundary: 10
inside_range: TRUE
no_change: 15
```

### Example: Monitor Used Disk Space on Windows

This example monitors the C drive on a Windows computer for used space. When the value of disk space used falls below 16 percent or exceeds 95 percent, the job completes.

```
insert_job: omd_win
job_type: OMD
machine: winagent
disk_drive: C
disk_format: PERCENT
lower_boundary: 16
upper_boundary: 95
inside_range: FALSE
disk_space: USED
```

### Example: Monitor Used Disk Space on i5/OS

This example monitors the system auxiliary storage pool on an i5/OS computer for used space. When 90 to 100 percent of the storage space is used, the job completes.

```
insert_job: omd_i5
job_type: OMD
machine:i5agent
disk_drive: /
disk_format: PERCENT
lower_boundary: 90
upper_boundary: 100
inside_range: TRUE
disk_space: USED
```

## encoding Attribute—Specify the Character Encoding of a File

The encoding attribute specifies the character encoding of the text file to be monitored.

**Note:** If you do not specify the encoding attribute in your job definition, the job monitors the file as US-ASCII (the default).

### Supported Job Type

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### Syntax

This attribute has the following format:

encoding: *char\_set\_name*

***char\_set\_name***

Specifies the name of the character set used to encode the data in the file.

**Default:** US-ASCII

**Limits:** Up to 42 characters

**Examples:** ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16

**Note:** The supported character sets depends on your operating system and JVM.

### Example: Monitor a Text File Encoded in ISO-8859-1

This example monitors a text file that contains data encoded in the ISO Latin Alphabet No. 1 (also named ISO-LATIN-1). The job checks the text file immediately and completes successfully if the specified string is found. If the string is not found, the job fails.

```
insert_job: textfile_job
job_type: OMTF
machine: monagt
text_file_name: /export/home/logs/transactions.log
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 1
upper_boundary: 50
encoding: ISO-8859-1
monitor_mode: NOW
```

## endpoint\_URL Attribute—Specify a Target Endpoint URL

The endpoint\_URL attribute specifies the target endpoint address URL in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

endpoint\_URL: *endpoint\_url*

***endpoint\_url***

Specifies the target endpoint address URL. In a published WSDL file, the URL defining the target endpoint address is found in the location attribute of the port's soap:address element.

**Limits:** Up to 256 characters; case-sensitive

**Note:** In a Web Service job, if you specify the WSDL\_URL attribute but not the endpoint\_URL attribute, you must specify both the service\_name and port\_name attributes. For the job to run successfully without the endpoint\_URL attribute, the agent must be running on the same computer as the application server such as WebLogic or JBoss. If you specify both the WSDL\_URL and endpoint\_URL attributes, then the service\_name and port\_name attributes are optional.

**Example: Specify the Target Endpoint Address URL for Getting Stock Quotes**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webserviceX.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type `string` in the return namespace `http://www.webserviceX.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## envvars Attribute—Specify a Job's Environment Variables

The envvars attribute specifies environment variables to define the local environment where the script, command, or batch file runs. You can modify existing environment variables or create your own.

### Supported Job Types

This attribute is optional for the following job types:

- [Command \(CMD\)](#) (see page 204)
- [PeopleSoft \(PS\)](#) (see page 249)
- [z/OS Regular \(ZOS\)](#) (see page 285)

### Syntax

This attribute has the following format:

`envvars: parm_name=parm_value[, parm_name=parm_value...]`

#### *parm\_name*

Defines the name of a new environment variable or specifies the name of an existing environment variable.

**Note:** Enclose names that contain spaces in quotation marks.

**Example:** "exceptions 1"

#### *parm\_value*

Specifies the value of the environment variable.

**Note:** Enclose values that contain spaces in quotation marks.

**Example:** "cmd1 cmd2 cmd3"

### Notes:

- You can specify multiple *parm\_name=parm\_value* combinations. Separate each combination with a comma.
- You can use multiple entries of envvars to define different sets of environment variables. For example, you can specify two sets of combinations using two envvars attributes as follows:  
`envvars: path1="c:\test". Path2="c:\prod" ...`
- Each *parm\_name=parm\_value* combination can be up to 1024 characters.
- Each entry of envvars can be up to 4096 characters.

- Programs, commands, batch files, and scripts can access environment variables using the getenv() function. Specify variable names for your environment as follows:
  - For UNIX scripts or commands to access environment variables, prefix the variable name with a dollar sign (\$).
  - For Windows batch files or programs to access environment variables, specify the variable name with a leading and trailing percent sign (%).
- The envvars attribute does not override Windows system variables. To reset Windows system-defined variables, set these variables in a batch file. For example, code the following to set variables TMP and TEMP in a batch file:
 

```
set TMP=d:\spool
set TEMP=d:\agent\temp
```
- The environment variables defined using the envvars attribute are processed before the job profile specified in the profile attribute. For more information about environment variables and how the environment for a job is sourced, see the *User Guide*.

#### **Example: Pass UNIX Environment Variables to a Script**

This example includes two envvars attributes that pass environment variables to a script and a third envvars attribute that defines the Present Working Directory (PWD). The parameter "user 1" is enclosed with quotation marks because it contains a space.

```
insert_job: unix_job
job_type: CMD
machine: unixprod
command: /home/scripts/pay
envvars: NAME="user 1"
envvars: JOB=PAYROLL
envvars: PWD=/usr/scripts/dailyrun
```

In this example, the pay script can reference these variables:

| Environment Variable | Value Passed          |
|----------------------|-----------------------|
| NAME                 | user 1                |
| JOB                  | PAYROLL               |
| PWD                  | /usr/scripts/dailyrun |

**Note:** If the parameter oscomponent.loginshell is set to true in the agent's agentparm.txt file, the agent invokes the user environment while running the script. To override the value of a shell variable that is already defined in the user login file, reassign a value to this variable in the script.

#### **Example: Define Two Environment Variables on Windows**

This example runs the processrun.exe command file on a Windows computer named winprod. The job uses two local environment variables named PATH and TEMP.

```
insert_job: cmd_job1
job_type: CMD
machine: winprod
command: "c:\cmds\processrun.exe"
envvars: PATH="c:\windows\system32"
envvars: TEMP="c:\temp"
```

## **exclude\_calendar Attribute—Define Days to Exclude a Job from Running**

The exclude\_calendar attribute identifies a custom calendar to use to determine the days of the week on which to exclude the job you are defining from running. You can use a standard calendar or an extended calendar as the exclude\_calendar.

Custom calendars are useful for complex date specification, such as when you need to exclude a job from running on the last business day of the month. Custom calendars can specify any number of dates on which to exclude associated jobs from running. Each calendar is stored as a separate uniquely named object in the database and can be associated with one or more jobs. The specified calendar must have previously been created using the autocal\_asc command.

When you specify an exclude\_calendar as the only date condition and the job has other implicit or explicit start conditions, the scheduler inspects the calendar before starting the job. If the current date is on the calendar, the job does not start and its status changes to INACTIVE. If the job's status changes to INACTIVE and it is in a box, the box job completes when all other conditions are satisfied. Also, if the job is a box job and its status changes to INACTIVE, all the jobs in the box change to INACTIVE.

**Note:** Times set in the start\_times and start\_mins attributes override times set in a custom calendar. The date\_conditions attribute controls the usage of this attribute.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`exclude_calendar: calendar_name`

#### *calendar\_name*

Defines the name of a custom calendar that was created with the `autocal_asc` command.

**Default:** No calendar is used.

**Limits:** Up to 30 alphanumeric characters

### Example: Specify the Exclusion Calendar for a Job

This example specifies the custom calendar HOLIDAY that defines days when the job cannot run.

```
date_conditions: y
exclude_calendar: HOLIDAY
```

## fail\_codes Attribute—Define Exit Codes to Indicate Job Failure

The `fail_codes` attribute defines which exit codes indicate job failure. For example, you can define the `fail_codes` attribute to be 40-50. If a job returns an exit code between 40 and 50, the job is considered to have failed.

**Note:** If you do not specify the `fail_codes` attribute in your job definition, the scheduling manager interprets any exit code other than (0) zero as job failure (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [Command \(CMD\)](#) (see page 204)
- [i5/OS \(I5\)](#) (see page 224)

## Syntax

This attribute has the following format:

`fail_codes: exit_codes`

**`exit_codes`**

Defines which exit codes indicate job failure. Specify a single exit code (for example, 4), a range of exit codes (for example, 0-9999), a list of multiple exit codes separated by commas, or a list of ranges separated by commas.

**Limits:** Up to 256 characters

**Default:** Any exit code other than 0 (zero) indicates job failure

**Examples:** -20--10,1,20-30,150

**Note:** If you specify multiple exit codes, enter the most specific codes first followed by the ranges.

## Example: Define a Range of Exit Codes to Indicate Job Failure

Suppose that you want a job named CMDJOB to run the procjob.exe file. The job is considered to have failed if the exit code is in the 40-50 range.

```
insert_job: CMDJOB
job_type: CMD
machine: winagt
command: "c:\temp\procjob.exe"
fail_codes: 40-50
```

## filter Attribute—Specify a Filter Using Regular Expression Logic

The filter attribute specifies a regular expression to use as a filter in a JMS Subscribe, JMX-MBean Subscribe, and HTTP job.

### Supported Job Types

This attribute is required for the [JMX-MBean Subscribe \(JMXSUB\)](#) (see page 240) job type if you specify Types in the filter\_type attribute.

This attribute is optional for the following job types:

- [HTTP](#) (see page 222)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)

### Syntax

This attribute has the following format:

- For the JMSSUB and HTTP job types:

filter: *filter*

- For the JMXSUB job type:

filter: *attribute* | *type*[,*attribute* | *type*...]

#### *filter*

Specifies a regular expression to use as a filter. In a JMS Subscribe job, this attribute filters messages from the topic or queue. In an HTTP job, this attribute filters the output of the invoked servlet. In a JMX-MBean Subscribe job, this attribute filters the notifications by attribute or by notification type.

**Example:** .+(CA)+.+

#### *attribute*

Identifies an attribute to monitor for change.

**Limits:** Case-sensitive

#### *type*

Identifies a notification type to monitor for.

**Limits:** Case-sensitive

**Example:** jmx.attribute.change (sends a notification every time an attribute changes)

**Notes:**

- The entire value can be up to 256 characters.
- To compose a regular expression, follow the rules for Java class `java.util.regex.Pattern`. You can find these rules on the Internet by searching for "java pattern".

**Example: Monitor a Queue on a WebLogic Application Server for a Message Matching Filter Criteria**

This example continuously monitors the queue named Queue (residing on WebLogic) for a message matching the filter criteria. The consumed messages from the queue are stored in the file `/export/home/user1/outputfile1`. The service provider's URL is `t3://172.24.0.0:7001`, where 172.24.0.0 is the IP address of the WebLogic Application server and 7001 is the ORB port.

```
insert_job: monitor
job_type: JMSSUB
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://172.24.0.0:7001"
connection_factory: ConnectionFactory
destination_name: Queue
continuous: Y
filter: abc\s...\\s[a-zA-Z]+\sFilter![\sa-zA-Z0-9]+
use_topic: FALSE
destination_file: /export/home/user1/outputfile1
j2ee_user: cyberuser
```

In this example, the regular expression used as the filter criteria can be defined as follows:

**abc\s...\\s[a-zA-Z]+\sFilter![\sa-zA-Z0-9]+**

**abc\s**

Specifies the text abc, followed by white space.

**...\\s**

Specifies any three characters, followed by white space.

**[a-zA-Z]+\s**

Specifies at least one letter, followed by white space.

**Filter![\sa-zA-Z0-9]+**

Specifies the text Filter!, followed by at least one of the following: white space or digit or lower case letter.

**Example:** abc vvv B Filter! 95

### Example: Monitor for a Change to a Specific MBean Attribute Using a Filter

Suppose that you want to monitor for a change to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. When the cachesize attribute changes, the job completes.

```
insert_job: change
job_type: JMXSUB
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
filter_type: Attributes
filter: cachesize
```

## filter\_type Attribute—Specify the Filter Type

The filter\_type attribute specifies whether to filter notifications by attribute or by notification type in a JMX-MBean Subscribe job.

**Note:** If you do not specify the filter\_type attribute in your job definition, the job filters notifications by attribute (the default).

### Supported Job Type

This attribute is optional for the [JMX-MBean Subscribe \(JMXSUB\) job type](#) (see page 240).

### Syntax

This attribute has the following format:

filter\_type: Attributes | Types

#### Attributes

Filters notifications by attribute. This is the default.

#### Types

Filters notifications by notification type.

**Note:** If you specify filter\_type: Types, you must also specify the filter attribute.

#### **Example: Monitor for a Change to a Specific MBean Attribute Using a Filter**

Suppose that you want to monitor for a change to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. When the cachesize attribute changes, the job completes.

```
insert_job: change
job_type: JMXSUB
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
filter_type: Attributes
filter: cachesize
```

#### **Example: Monitor for Changes to Any MBean Attribute**

Suppose that you want to set up continuous monitoring for changes to any attribute of the MBean named Config. Each time an attribute changes, an alert is written to the scheduler log file.

```
insert_job: change
job_type: JMXSUB
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
continuous: Y
filter_type: Types
filter: jmx.attribute.change
```

## **finder\_name Attribute—Specify a Finder Method**

The finder\_name attribute specifies the name of the finder method in an Entity Bean job. You require the finder method to update the property values of an existing entity bean or to remove an entity bean from the database. To find the entity bean, the agent uses the bean's Java Naming and Directory Interface (JNDI) name along with its finder method.

### **Supported Job Type**

When the operation type is UPDATE or REMOVE, this attribute is required for the [Entity Bean \(ENTYBEAN\) job type](#) (see page 214).

### **Syntax**

This attribute has the following format:

```
finder_name: finder_method
```

***finder\_method***

Specifies the name of the finder method.

**Limits:** Up to 256 characters; case-sensitive

**Example: Update the Phone Number for the Acme Company in a Database**

Suppose that you want to update the phone number for the Acme company to 800-555-0199. The customer entity bean stores the customer ID and phone number. The primary key for the customer is the customer ID. To find the entity bean, the job uses the Acme's customer ID. When the job runs, the Acme company's phone number is changed.

```
insert_job: update
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: UPDATE
method_name: changephone
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
modify_parameter: String="800-555-0199"
```

## finder\_parameter Attribute—Specify Finder Parameters

The `finder_parameter` attribute specifies the finder parameters in an Entity Bean job. You specify finder parameters to update the property values of an existing entity bean or to remove an entity bean from the database.

**Supported Job Type**

When the operation type is UPDATE or REMOVE, this attribute is required for the [Entity Bean \(ENTYBEAN\) job type](#) (see page 214).

**Syntax**

This attribute has the following format:

```
finder_parameter: type=value | payload_job=job_name[, type=value | payload_job=job_name...]
```

***type=value***

Specifies the Java class and String value for the parameter.

**Limits:** Up to 1024 characters; case-sensitive

**Examples:** String="5:00PM", String[2]="winnt1,winnt2", double=2000

**Note:** If the package of the Java class is not specified, java.lang is assumed.

***payload\_job=job\_name***

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

**Notes:**

- You can define up to 64 finder parameters using one `finder_parameter` attribute or multiple entries of `finder_parameter`.
- In each `finder_parameter` entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. Specify the parameters in the same order as they appear in the `finder` method. The job fails if the parameters are listed in the wrong order.

**Example: Update the Phone Number for the Acme Company in a Database**

Suppose that you want to update the phone number for the Acme company to 800-555-0199. The customer entity bean stores the customer ID and phone number. The primary key for the customer is the customer ID. To find the entity bean, the job uses the Acme's customer ID. When the job runs, the Acme company's phone number is changed.

```
insert_job: update
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: UPDATE
method_name: changephone
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
modify_parameter: String="800-555-0199"
```

## ftp\_command—Specify Site-specific FTP Commands

The ftp\_command attribute specifies the commands to execute prior to file transfer. You can use this attribute to send site-specific FTP commands to FTP servers.

### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

`ftp_command: command[,command...]`

#### *command*

Defines a command that is to be executed prior to file transfer.

**Limits:** Up to 4096 characters

#### Notes:

- You can specify multiple commands. Separate each command with a comma.
- You can use multiple entries of ftp\_command to define different sets of commands. For example, you can specify two sets of commands using two ftp\_command attributes as follows:

```
ftp_command: command1,command2
ftp_command: command3,command4
```
- Each entry of ftp\_command can be up to 4096 characters.
- This attribute does not support all FTP commands. Use this attribute only to set up the FTP connection so that the data transfers correctly. For example, you can specify this attribute to run the LOCSITE command.

#### Example: Send FTP Commands to an FTP Server

This example sends two FTP commands to the FTP server prior to transferring a file.

```
insert_job: CYBJK.FTP
job_type: FTP
machine: ftppagent
ftp_server_name: ftp.ca.com
ftp_server_port: 21
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: B
ftp_compression: 8
ftp_remote_name: /pub/cazip.exe
ftp_local_name: /tmp/bla
ftp_command: locsite blksize=FB
ftp_command: locsite automount
owner: user1@ftp.ca.com
```

## ftp\_compression Attribute—Specify the Data Compression Level

The `ftp_compression` attribute specifies the compression level for FTP data transfer.

**Note:** If you do not specify the `ftp_compression` attribute in your job definition, the data is compressed using the level set in the `ftp.data.compression` parameter on the agent FTP client. This parameter is automatically set to 0 by default (no data compression).

#### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

#### Syntax

This attribute has the following format:

`ftp_compression: level`

***level***

Specifies the data compression level.

**Limits:** 0-9 (0 is no data compression and 9 is the highest data compression)

**Default:** 0

**Notes:**

- To use the compression feature, both the FTP client and the FTP server must run on the agent software. If this value is specified and the FTP server or the FTP client does not run on the agent, the data will be transferred without compression.
- On the agent FTP client, your agent administrator can specify the default compression level for all FTP jobs by setting the ftp.data.compression parameter in the agent's agentparm.txt file.
- The ftp\_compression attribute overrides the default compression level specified in the agent's agentparm.txt.

**Example: Compress a File**

The local computer in this example has an agent running as an FTP client. The remote server has an agent running as an FTP server. Both computers operate on a low bandwidth network.

The following FTP job downloads a large file named largefile.txt from the remote server to the FTP client. The computers are on a low bandwidth network, so the data is compressed at compression level 3.

```
insert_job: FTPJOB
job_type: FTP
machine: ftppagent
ftp_server_name: aixunix
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_compression: 3
ftp_remote_name: /files_to_download/largefile.txt
ftp_local_name: "c:\downloaded_files\largefile.txt"
owner: ftpuser@aixunix
```

## ftp\_local\_name Attribute—Specify Local Filenames within an FTP Job

The ftp\_local\_name attribute specifies the name of one or more files on the agent computer involved in an FTP transfer.

### Supported Job Type

This attribute is required for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

`ftp_local_name: file_name[; file_name...]`

#### *file\_name*

Specifies the destination of the file (if downloading) or the source location of the file (if uploading).

**Limits:** Case-sensitive

#### UNIX/Windows:

- If you are downloading a file, you must specify the full path and file name.
- If you are downloading multiples files using wildcards in the ftp\_remote\_name, you must specify the full path to a directory name.
- If you are uploading files, you can use wildcards in the file name. The asterisk (\*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- You cannot use wildcards in the path.

#### i5/OS:

- To specify a file in the root file system, use UNIX path and file formats.
- To specify a \*FILE object in QSYS, use the following format:

`/QSYS.LIB/libraryname.LIB/objectname.FILE/membername.MBR`

**Notes:**

- The entire value can be up to 256 characters.
- If a wildcard is used in a local file name for upload, the remote file name (the target) must refer to a directory. A wildcard transfer is equivalent to an mget transfer using an FTP client.
- You can specify multiple files. Separate each file name with a semi-colon (the default delimiter to separate multiple files). The number of files specified in the ftp\_local\_name and ftp\_remote\_name attributes must match.
- Your agent administrator can change the default delimiter that separates file names by setting the ftp.client.separator parameter in the agent's agentparm.txt file.

**Example: Download a Single File**

The FTP job in this example downloads an ASCII file named textfile from the remote UNIX server to the /export/home/ftpfiles/ftpdata/Folder directory on the agent computer.

```
insert_job: FTPDL_SINGLE
job_type: FTP
machine: ftpagent
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: u1/files/ftpdata/textfile
ftp_local_name: /export/home/ftpfiles/ftpdata/Folder/textfile
owner: ftpuser@hprsupp
```

### **Example: Download Multiple Local Files**

The FTP job in this example downloads multiple ASCII files from a remote UNIX server. Note that the number of files listed in the `ftp_remote_name` and `ftp_local_name` attributes must match.

```
insert_job: DOWNLOAD_MULTIPLE
job_type: FTP
machine: RAGENT
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name:
/u1/files/scripts/echo;/u1/files/scripts/echo1;/u1/files/scripts/echo2
ftp_local_name:
/export/home/ftpfiles/ftpdata/echo;/export/home/ftpfiles/ftpdata/echo_1;/export/home/ftpfiles/ftpdata/echo_2
owner: ftpuser@hprsupp
```

### **Example: Upload Files to a Directory Using a Wildcard**

If a wildcard is used in a local file name for upload, the remote file name (the target) must refer to a directory. In this example, the remote file name is specified as the directory `/tmp`.

```
insert_job: FTPUPLOAD
job_type: FTP
machine: ftppagent
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: UPLOAD
ftp_transfer_type: A
ftp_remote_name: /tmp
ftp_local_name: /export/home/ftpfiles/ftpdata/text*
owner: ftpuser@hprsupp
```

**Note:** For Windows examples, substitute a Windows path in the `ftp_local_name` attribute and enclose the path in quotation marks.

### Example: Download a QSYS.LIB EBCDIC-encoded File

This example downloads an EBCDIC-encoded file in the QSYS file system from an i5/OS server to another i5/OS system. The file names are specified in the path format.

```
insert_job: FTP_EBCDIC
job_type: FTP
machine: I5AGENT
ftp_server_name: i5agent
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: E
ftp_local_name: /QSYS.LIB/ESPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
ftp_remote_name: /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
owner: ftpuser@i5agent
```

## ftp\_local\_user Attribute—Specify the Local User ID on the Agent

The ftp\_local\_user attribute specifies the local user ID on agent machine. The job uses this user ID to determine the level of access for the file on the local system. For example, if ftp\_transfer\_direction attribute is set to DOWNLOAD and ftp\_local\_user is set to JDOE, the downloaded file is created with the JDOE as the owner.

#### Notes:

- If you do not specify the ftp\_local\_user attribute in your job definition, the job uses the user who inserted the job (the default).
- The ftp\_local\_user attribute differs from the owner attribute. The owner attribute specifies the FTP user ID that is used to log in to the FTP server to run the job. The FTP local user is used to check whether the specified file can be accessed.

#### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

#### Syntax

This attribute has the following format:

```
ftp_local_user: user
```

*user*

Specifies the the local user ID on the agent machine.

**Limits:** Up to 80 characters

**Default:** The user who inserted the job

## ftp\_remote\_name Attribute—Specify Remote Filename within an FTP Job

The ftp\_remote\_name attribute specifies the name of one or more files on a remove server involved in an FTP transfer.

### Supported Job Type

This attribute is required for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

`ftp_remote_name: file_name[; file_name...]`

#### *file\_name*

Specifies the source location of the file (if downloading) or the destination of the file (if uploading).

**Limits:** Case-sensitive

#### UNIX/Windows:

- If you are uploading a file, you must specify the full path and file name.
- If you are uploading multiple files by specifying a file name containing wildcard characters for the ftp\_local\_name attribute, you must specify the full path of a directory where the files will be stored.
- If you are downloading files, you can use wildcards in the file name. The asterisk (\*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- You cannot use wildcards in the path.
- On UNIX, if you want to use a Windows file as a remote file, you must use a forward slash at the beginning of the path statement and between the directories and file name, as shown in the following example:

`ftp_remote_name: "/C:/Temp/textfile"`

#### i5/OS:

- To specify a file in the root file system, use UNIX path and file formats.
- To specify a \*FILE object in QSYS, use the following format:

`/QSYS.LIB/Libraryname.LIB/Objectname.FILE/membername.MBR`

**Notes:**

- The entire value can be up to 256 characters.
- If a wildcard is used in a remote file name for download, the local file name (the target) must refer to a directory. A wildcard transfer is equivalent to an mget transfer using an FTP client.
- You can specify multiple files. Separate each file name with a semi-colon (the default delimiter to separate multiple files). The number of files specified in the ftp\_local\_name and ftp\_remote\_name attributes must match.
- Your agent administrator can change the default delimiter that separates file names by setting the ftp.client.separator parameter in the agent's agentparm.txt file.

**Example: Download a Single File**

The FTP job in this example downloads an ASCII file named textfile from the remote UNIX server to the /export/home/ftpfiles/ftpdata/textfile\_dn directory on the agent computer.

```
insert_job: DOWNLOAD_SINGLE
job_type: FTP
machine: RAGENT
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/files/ftpdata/textfile
ftp_local_name: /export/home/ftpfiles/ftpdata/textfile_dn/textfile
owner: testuser@hprsupp
```

### **Example: Download Multiple Local Files**

The FTP job in this example downloads multiple ASCII files from a remote UNIX server. Note that the number of files listed in the `ftp_remote_name` and `ftp_local_name` attributes must match.

```
insert_job: DOWNLOAD_MULTIPLE
job_type: FTP
machine: RAGENT
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name:
/u1/files/scripts/echo;/u1/files/scripts/echo1;/u1/files/scripts/echo2
ftp_local_name:
/export/home/ftpfiles/ftpdata/echo;/export/home/ftpfiles/ftpdata/echo_1;/export/home/ftpfiles/ftpdata/echo_2
owner: ftpuser@hprsupp
```

### **Example: Download Files to a Directory Using a Wildcard**

If a wildcard is used for the remote file name, the corresponding local file name (the target) must refer to a directory. In this example, the directory is WC.

```
insert_job: FTP_WILDCARD
job_type: FTP
machine: ftppagent
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/files/scripts/r*
ftp_local_name: /export/home/ftpfiles/ftpdata/WC
owner: test@hprsupp
```

**Example: Upload a File to a Remote Windows Computer**

In this example, a file named `textfile` is uploaded from a local UNIX computer and copied to a remote Windows computer. Both locations include a complete path in the `ftp_remote_name` and `ftp_local_name` attributes. The remote Windows computer uses an agent as the FTP server. Its IP address is 172.16.0.0.

```
insert_job: FTP_REMOTE
job_type: FTP
machine: unixagent
ftp_server_name: 172.16.0.0
ftp_server_port: 123
ftp_transfer_direction: UPLOAD
ftp_transfer_type: A
ftp_remote_name: "/C:/Temp/textfile"
ftp_local_name: /export/home/ftpfiles/ftpdata/textfile
owner: test@172.16.0.0
```

**Note:** To transfer a file from a Windows computer, substitute a Windows path in the `ftp_local_name` attribute and enclose the entire path in quotation marks.

**Example: Download a QSYS.LIB EBCDIC-encoded File**

This example downloads an EBCDIC-encoded file in the QSYS file system from an i5/OS server to another i5/OS system. The file names are specified in the path format.

```
insert_job: FTP_EBCDIC
job_type: FTP
machine: I5AGENT
ftp_server_name: i5agent
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: E
ftp_local_name: /QSYS.LIB/ESPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
ftp_remote_name: /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
owner: ftpuser@i5agent
```

## ftp\_server\_name Attribute—Specify FTP Server Name within an FTP Job

The ftp\_server\_name attribute specifies the name of the FTP server in an FTP job.

### Supported Job Type

This attribute is required for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

ftp\_server\_name: *server\_address*

#### *server\_address*

Specifies the DNS name or IP address of a remote server.

**Limits:** Up to 256 characters; case-sensitive

**Example:** 172.24.36.107 (IPv4) or 0:0:0:0:FFFF:192.168.00.00 (IPv6)

### Example: Download a File from a UNIX Computer

This example downloads a UNIX file from the hpsupport server with port 5222.

```
insert_job: FTPT1A
job_type: FTP
machine: ftppagent
ftp_server_name: hpsupport
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/qatest/ftpdata/textfile
ftp_local_name: /export/home/ftpfiles/test1
owner: ftpuser@hpsupport
```

**Example: Download a File from a UNIX Computer to a Windows Computer**

The FTP job in this example downloads a file from a UNIX server with IP address 172.16.0.0 and port 21 to a Windows computer.

```
insert_job: FTPT1A
job_type: FTP
machine: ftpagent
ftp_server_name: 172.16.0.0
ftp_server_port: 21
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/ftpdata/textfile
ftp_local_name: "c:\ftpfiles"
owner: ftpuser@172.16.0.0
```

**Example: Download a QSYS.LIB EBCDIC-encoded File**

This example downloads an EBCDIC-encoded file in the QSYS file system from an i5/OS server to another i5/OS system. The file names are specified in the path format.

```
insert_job: FTP_EBCDIC
job_type: FTP
machine: I5AGENT
ftp_server_name: i5agent
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: E
ftp_local_name: /QSYS.LIB/ESPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
ftp_remote_name: /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
owner: ftpuser@i5agent
```

## ftp\_server\_port Attribute—Specify the Server Port of a Remote FTP Server

The ftp\_server\_port attribute specifies the port number of the remote FTP server involved in an FTP file transfer.

### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

ftp\_server\_port: *port*

***port***

Specifies the port number of the remote server.

**Limits:** 0-65535

**Example:** 5222

### Example: Specify a Server Port Number

The FTP job in this example downloads a file from a UNIX server with IP address 172.16.0.0 and port 21 to a Windows computer.

```
insert_job: FTPT1A
job_type: FTP
machine: ftppagent
ftp_server_name: 172.16.0.0
ftp_server_port: 21
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/ftpdata/textfile
ftp_local_name: "c:\ftpfiles"
owner: ftpuser@172.16.0.0
```

## ftp\_transfer\_direction Attribute—Specify Transfer Direction within an FTP Job

The ftp\_transfer\_direction attribute specifies the file transfer direction in an FTP job.

**Note:** If you do not specify the ftp\_transfer\_direction attribute in your job definition, the job downloads the data (the default).

### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

ftp\_transfer\_direction: UPLOAD | DOWNLOAD

#### UPLOAD

Transfers files from the agent computer to the remote server.

#### DOWNLOAD

Transfers files from the remote server to the agent computer. This is the default.

### Example: Upload Files

If a wildcard is used in a local file name for upload, the remote file name (the target) must refer to a directory. In this example, the remote file name is specified as the directory /tmp.

```
insert_job: FTPUPLOAD
job_type: FTP
machine: ftppagent
ftp_server_name: hprsupp
ftp_server_port: 5222
ftp_transfer_direction: UPLOAD
ftp_transfer_type: A
ftp_remote_name: /tmp
ftp_local_name: /export/home/ftpfiles/ftpdata/text*
owner: ftpuser@hprsupp
```

**Note:** For Windows examples, substitute a Windows path in the ftp\_local\_name attribute and enclose the path in quotation marks.

## ftp\_transfer\_type Attribute—Specify Data Code Type within an FTP Job

The ftp\_transfer\_type attribute specifies the type of data involved in an FTP transfer (binary, ASCII, or EBCDIC).

**Note:** If you do not specify the ftp\_transfer\_type attribute in your job definition, the job performs a binary data transfer (the default).

### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

ftp\_transfer\_type: A |B | E

#### A

Indicates an ASCII transfer.

#### i5/OS:

- If the ASCII file to be transferred already exists on the target computer, the file is written using the encoding of the existing file.
- If the file does not exist, the file is written using the ASCII CCSID (Coded Character Set Identifier) defined on the agent. The default is 819.

#### B

Indicates a binary transfer. This is the default.

#### E

Indicates an EBCDIC transfer.

**Note:** E applies to CA WA Agent for i5/OS only. If an EBCDIC file to be transferred already exists on the target computer, the file is written using the encoding of the existing file. If the file does not exist, the file is written using the EBCDIC CCSID (Coded Character Set Identifier) defined on the agent. The default is 37.

### Example: Download a Binary File on UNIX

The FTP job in this example uses a binary transfer.

```
insert_job: FTpbinary
job_type: FTP
machine: unixagent
ftp_server_name: hpunix
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: B
ftp_remote_name: /u1/qatest/ftpdata/binaryfile
ftp_local_name: /export/home/qatest/ftpdata/transf.bin
owner: test@hpunix
```

### Example: Upload an ASCII-encoded File in the Root File System from an i5/OS System to a UNIX System

This example uploads the textdoc file in the root file system from an i5/OS system and copies it to a UNIX system.

```
insert_job: UPLOAD_i5
job_type: FTP
machine: I5AGENT
ftp_server_name: hpunix
ftp_server_port: 5222
ftp_transfer_direction: UPLOAD
ftp_transfer_type: A
ftp_remote_name: /u1/ftpfiles/ftpdata/textdoc
ftp_local_name: /home/ftp/textdoc
owner: test@hpunix
```

### Example: Download a QSYS EBCDIC-encoded File

This example downloads an EBCDIC-encoded file named DATAFILE in the QSYS file system from an i5/OS system to another i5/OS system. The file names are specified in the path format.

```
insert_job: EBCDIC_FILE
job_type: FTP
machine: I5AGENT
ftp_server_name: i5agent
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_transfer_type: E
ftp_remote_name: /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
ftp_local_name: /QSYS.LIB/ESPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
owner: test@i5agent
```

## ftp\_use\_SSL Attribute—Specify SSL FTP or Regular FTP

The ftp\_use\_SSL attribute specifies whether to transfer the data with Secure Sockets Layer (SSL) communication or regular communication.

**Note:** If you do not specify the ftp\_use\_SSL attribute in your job definition, the data is transferred using the communication type set in the ftp.client.ssl parameter on the agent FTP client. This parameter is automatically set to false by default (regular communication is used).

### Supported Job Type

This attribute is optional for the [FTP job type](#) (see page 220).

### Syntax

This attribute has the following format:

`ftp_use_SSL: TRUE | FALSE`

#### **TRUE**

Transfers data using SSL FTP.

#### **FALSE**

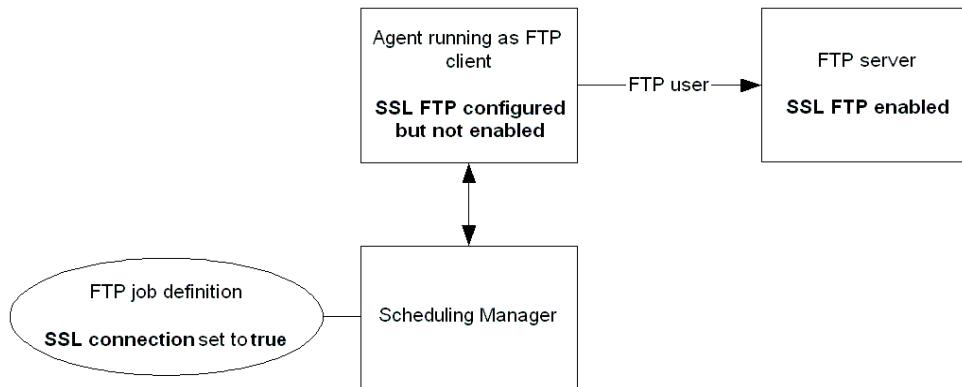
Transfers data using regular FTP. This is the default.

### Notes:

- To transfer data using SSL, check for the following:
  - The FTP server must have SSL FTP enabled.
  - The FTP client must have SSL FTP configured (SSL FTP can be enabled or disabled).
- The agent administrator can enable or disable SSL on the agent FTP client using the ftp.client.ssl parameter in the agent parameter file (agentparm.txt) as follows:
  - If the agent FTP client has SSL FTP enabled, all FTP jobs on that agent automatically use SSL FTP.
  - If the agent FTP client does not have SSL FTP enabled, all FTP jobs on that agent automatically use regular FTP.
- The ftp\_use\_ssl attribute overrides the ftp.client.ssl parameter specified in the agent's agentparm.txt.
- If the FTP client has SSL FTP enabled, but the FTP server does not, you must set the ftp\_use\_SSL attribute to FALSE. Otherwise, the job will fail.

**Example: Upload a File from a Local Computer to a Remote Windows Server Using SSL FTP**

In this example, the agent runs on a local computer as an FTP client and has SSL FTP configured but not enabled. The remote Windows server has SSL FTP configured and enabled.



The following FTP job uploads the filename.txt file from the agent FTP client to the c:\uploaded\_files directory on a remote Windows server. The ftp\_use\_SSL attribute is set to TRUE to transfer the file securely. Although the agent FTP client does not have SSL FTP enabled, the file will be uploaded using SSL FTP because the configuration requirements are met (the agent FTP client has SSL FTP configured and the FTP server has SSL FTP enabled).

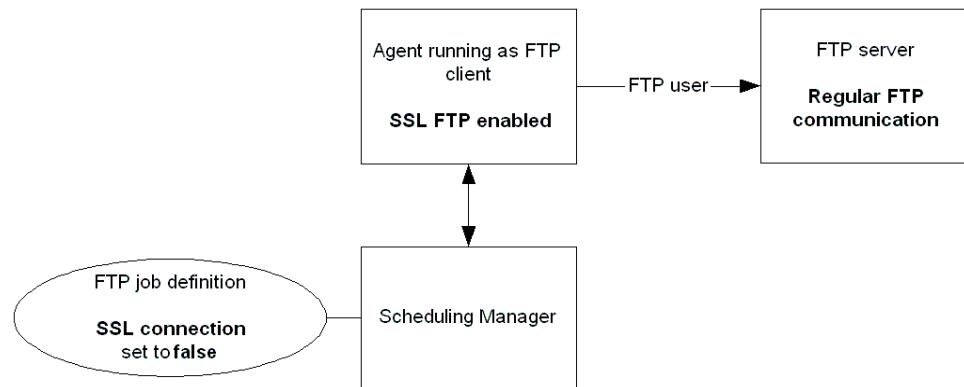
```

insert_job:FTP_UPLOAD
job_type: FTP
machine: winagent
owner: user1@winserver
ftp_server_name: winserver
ftp_server_port: 21
ftp_transfer_direction: UPLOAD
ftp_use_SSL: TRUE
ftp_remote_name: "c:\uploaded_files\filename.txt"
ftp_local_name: "d:\files_to_upload\filename.txt"

```

**Example: Download a File from a Remote UNIX Server That Does Not Support SSL FTP to a Local Computer That Supports SSL FTP**

In this example, the agent runs on a local computer as an FTP client and has SSL FTP enabled (all FTP jobs on the agent computer run using SSL FTP by default). The remote UNIX server does not support SSL FTP.



The following FTP job downloads the filename.txt file from the remote UNIX server to the d:\downloaded\_files directory on the local computer. Because the FTP server does not support SSL FTP, the ftp\_use\_SSL attribute is set to FALSE so the job does not fail.

```
insert_job: FTP_DOWNLOAD
job_type: FTP
machine: winagent
owner: user1@hpunix
ftp_server_name: hpunix
ftp_server_port: 5222
ftp_transfer_direction: DOWNLOAD
ftp_use_SSL: FALSE
ftp_remote_name: /files_to_download/filename.txt
ftp_local_name: "d:\downloaded_files\filename.txt"
```

## group Attribute—Associate a Job with a Group

The group attribute associates a job with a specific group so users can classify, sort, and filter jobs by group name.

You can use the group attribute with the sendevent, job\_depends, and autorep commands. For example, you could use a single sendevent command to start all of the jobs associated with a specific group.

Consider the following when you use the group attribute:

- Users with execute permission can add new group attributes to a job definition.
- You can only associate a job with one group at a time.
- When a job is associated with a group and an application (using the application attribute), CA Workload Automation AE assumes that the application attribute is a subset of the group attribute.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

group: *group\_name*

***group\_name***

Defines the name of the group with which to associate the job. You can only associate a job with one group, but a group may have many jobs associated with it.

**Limits:** Up to 64 characters; valid characters are a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-); do not include embedded spaces or tabs

### Example: Associate a Job with a Group

This example associates a job with the group year\_end\_reports.

group: year\_end\_reports

## heartbeat\_interval Attribute—Set the Monitoring Frequency for a Job

The heartbeat\_interval attribute defines how often (in minutes) a Command job issues a heartbeat. Heartbeats are signals that monitor the progress of the application the job runs.

You can program the application to send heartbeats to the agent that starts the job. The agent listens for each heartbeat and sends a HEARTBEAT event to the CA Workload Automation AE event servers. Finally, the CA Workload Automation AE scheduler checks that the HEARTBEAT event has occurred.

If the job does not send a heartbeat to the agent during the specified interval, the agent generates a HEARTBEAT alarm.

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

### Syntax

This attribute has the following format:

heartbeat\_interval: *mins*

#### *mins*

Defines the frequency (in minutes) with which the job sends a heartbeat. The heartbeat is a signal that helps the scheduling manager monitor the progress of a job.

**Limits:** 0-2,147,483,647

**Default:** 0 (the agent does not listen for heartbeats)

### UNIX Notes:

- You must configure how often the CA Workload Automation AE scheduler checks for heartbeats. For more information about configuring the scheduler on UNIX to check for heartbeats, see the *Administration Guide*.
- To send a heartbeat from a C program, call the routine in the following file:  
\$AUTOSYS/code/heartbeat.c
- To send a heartbeat from a Bourne shell script, run the code in the following file:  
\$AUTOSYS/code/heartbeat.sh

**Windows Notes:**

- You must configure how often the CA Workload Automation AE scheduler check for heartbeats. For more information about configuring the scheduler on Windows to check for heartbeats, see the CA Workload Automation AE Administrator *Online Help*.
- To send a heartbeat from a C program, call the routine in the following file:  
%AutoSys%\code\testheart.c

**Example: Issue a Heartbeat Every Two Minutes**

This example defines a Command job that runs a UNIX script named backup. The job issues a heartbeat every two minutes to the agent that started the job.

```
insert_job: unix_script
job_type: CMD
machine: unixagent
command: /usr/common/backup
heartbeat_interval: 2
```

## i5\_action Attribute—Specify Whether to Run a Program or Issue a Command

The i5\_action attribute specifies whether to run a program or issue a command on an i5/OS system.

**Note:** The default value is COMMAND. If you do not specify the i5\_action attribute in your job definition, the job interprets the i5\_name value to be a command.

**Supported Job Type**

This attribute is optional for the [i5/OS job type](#) (see page 224).

**Syntax**

This attribute has the following format:

i5\_action: RUN\_PROGRAM | RUN\_PROG\_IN\_FILE | COMMAND

### **RUN\_PROGRAM**

Runs a program on an i5/OS system.

### **RUN\_PROG\_IN\_FILE**

Runs a program from a database file member on an i5/OS system. This is similar to running a shell script on a UNIX system or a batch file on a Windows system.

### **COMMAND**

Issues a command on an i5/OS system when the i5/OS job runs. This is the default.

#### **Example: Run a Program on an i5/OS System**

This example runs the program named PARAM under the ASYSOPR user ID. The program takes three parameters in the following order: ABC, C1, and P.

```
insert_job: i5job_runprog
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PARAM
i5_params: "ABC C1 P"
owner: ASYSOPR
```

#### **Example: Run a Command on an i5/OS System**

This example issues the DSPLIBL command to display the job's library list. When the job completes, you can view the library list in the job's spool file.

```
insert_job: i5job_displib
job_type: I5
machine: i5agent
i5_action: COMMAND
i5_name: DSPLIBL
i5_lib: *LIBL
i5_curr_lib: AGTLIB
i5_job_name: PAYJOBA
i5_job_desc: PRDLIB/PRDOPR
i5_job_queue: ABCSYS/PAYROLL
```

## i5\_cc\_exit Attribute—Specify the Type of Exit Code to Return

The i5\_cc\_exit attribute specifies the type of exit code returned by an i5/OS job. The exit code can be the job's ending severity code, the return code of an ILE program or module, or the return code of an OPM program.

**Note:** The default type is \*SEVERITY. If you do not specify the i5\_cc\_exit attribute in your job definition, the job sends the ending severity code as the exit code.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

i5\_cc\_exit: \*SEVERITY | \*USER | \*PROGRAM

#### \*SEVERITY

Sends the job's ending severity code. This is the default.

#### **\*USER**

Sends the return code of an ILE program or module. For example, if your job runs an ILE C or ILE RPG program that contains an exit or return statement, that return code is sent as the exit code.

**Note:** If the program does not set a return code, the i5 system returns a 0 (zero), which is sent as the exit code. By default, an exit code of zero (0) is interpreted as job success.

#### **\*PROGRAM**

Sends the return code of an OPM program. For example, if your job runs an OPM RPG or OPM COBOL program that contains an exit or return statement, that return code is sent as the exit code.

**Note:** If the program does not set a return code, the i5 system returns a 0 (zero), which is sent as the exit code. By default, an exit code of zero (0) is interpreted as job success.

**Notes:**

- Ensure that you specify the appropriate exit code type for the program you are running. If you specify the wrong exit code type (for example, you specify \*PROGRAM for an ILE program), the agent sends 0 (zero) as the exit code. By default, CA Workload Automation AE interprets exit codes of 0 (zero) as job success.
- If you specify \*PROGRAM or \*USER, the agent adds a custom message before the final completion message in the job's spool file (job log). This message is required to track the program's return code.

**Example: Send a C Program's Return Code as the Job's Exit Code**

This example runs a C language program named SALARY. The agent sends the SALARY program's return code to CA Workload Automation AE.

```
insert_job: i5job_returnC
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: SALARY
i5_cc_exit: *USER
```

**Example: Send an OPM COBOL Program's Return Code as the Job's Exit Code**

This example runs an OPM COBOL program named PAYROLL. The agent sends the PAYROLL program's return code to CA Workload Automation AE.

```
insert_job: i5job_returnOPM
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYROLL
i5_cc_exit: *PROGRAM
```

## i5\_curr\_lib Attribute—Specify the Name of the Current Library

The i5\_curr\_lib attribute specifies the name of the current library to use when running the job.

**Note:** If you do not specify a current library, the agent uses the current library of the user profile the agent runs under.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following formats:

```
i5_curr_lib: current_library
i5_curr_lib: /QSYS.LIB/current_library.LIB
current_library
```

Specifies the current library for the job.

**Note:** The entire value can be up to 46 characters; it cannot contain delimiters (such as spaces).

### Example: Specify the Current Library for a Job

This example runs a program named MFGDATA on an i5/OS system. The current library for the job is PAY1, and the job description is DFTJOBD in library CYB1.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: MFGDATA
i5_curr_lib: PAY1
i5_job_desc: CYB1/DFTJOBD
```

## i5\_job\_desc Attribute—Specify the Job Description

The i5\_job\_desc attribute specifies the name of the job description for the Control Language (CL) program that the job submits.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following formats:

*i5\_job\_desc: description*

*i5\_job\_desc: library/description*

*i5\_job\_desc: /QSYS.LIB/library.LIB/description.JOB*

#### *library*

Specifies the name of the library that contains the job description. The value must be a valid i5/OS library name.

#### *description*

Specifies the name of the job description for the program submitted. The value must be a valid i5/OS job description name.

**Limits:** Up to 10 characters; the first character must be one of the following: A-Z, \$, #, or @; the second to tenth characters can be A-Z, 0-9, \$, #, \_, ., or @

### Notes:

- The entire value can be up to 46 characters; it cannot contain delimiters (such as spaces).
- If you do not specify the library name, the i5/OS system will search for it using the library list for the job.
- Your agent administrator can specify a default job description for all i5/OS jobs by setting the os400.default.jobdbname parameter in the agent's agentparm.txt file.
- The i5\_job\_desc attribute overrides the default job description specified in the agent's agentparm.txt.

**Example: Specify the Library Name and Description for a Job**

This example runs a program named MFGDATA on an i5/OS system. The current library for the job is PAY1, and the job description is DFTJOBD in library CYB1.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: MFGDATA
i5_curr_lib: PAY1
i5_job_desc: CYB1/DFTJOBD
```

## i5\_job\_name Attribute—Specify the Job Name

The i5\_job\_name attribute specifies the name for an i5/OS job. This is the name displayed in the job queue.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

```
i5_job_name: job_name
job_name
```

Specifies the name of the i5/OS job.

**Limits:** Up to 10 characters; the first character must be one of the following: A-Z, \$, #, or @; the second to tenth characters can be A-Z, 0-9, \$, #, \_, ., or @

**Example: Specify the Name of an i5/OS Job**

This example defines an i5/OS job that runs a program named PROCPROG. The name of this job that is displayed in the job queue of the i5/OS system is PROG1.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PROCPROG
i5_job_name: PROG1
```

## i5\_job\_queue Attribute—Specify the Job Queue for a Program

The i5\_job\_queue attribute specifies the i5/OS job queue for the submitted program.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following formats:

*i5\_job\_desc: jobqueue*

*i5\_job\_desc: library/jobqueue*

*i5\_job\_desc: /QSYS.LIB/library.LIB/jobqueue.JOBQ*

#### *library*

Specifies the name of the library that contains the job queue. The value must be a valid i5/OS library name.

#### *job\_queue*

Specifies the name of the job queue for the submitted program. The value must be a valid i5/OS job queue name.

### Notes:

- The entire value can be up to 46 characters; it cannot contain delimiters (such as spaces).
- If you do not specify the library name, the i5/OS system will search for it using the library list for the job.
- Your agent administrator can specify a default job queue for all i5/OS jobs by setting the os400.default.jobqname parameter in the agent's agentparm.txt file.
- The i5\_job\_desc attribute overrides the default job queue specified in the agent's agentparm.txt.

**Example: Specify the Library Name and Job Queue for an i5/OS Job**

This example runs a program named PAYLOAD on an i5/OS system. The current library for the job is PAY1, and the job queue is JQUEUE in library QBASE.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYLOAD
i5_curr_lib: PAY1
i5_job_queue: /QYS.LIB/QBASE.LIB/JQUEUE.JOBQ
```

## i5\_lda Attribute—Specify the Data for the Local Data Area

The i5\_lda attribute specifies data for the local data area in an i5/OS job. The local data area is a temporary 1024-byte storage area that exists for the duration of the job. You can use the local data area to pass data to the job and to other programs that run as part of the job.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

i5\_lda: "data" | X'hh...'

#### "*data*"

Specifies the data in character format.

**Limits:** Up to 1024 characters; case-sensitive

**Example:** "My data"

#### X'*hh...*'

Specifies the data in hexadecimal format, with each pair of hexadecimal digits representing one byte of data.

**Limits:** Up to 2051 characters

**Example:** X'C1C2C340C1C2C3'

#### Example: Specify Data for the Local Data Area in Hexadecimal Format

This example defines an i5/OS job with data for the local data area. When the job is submitted, the agent initializes the local data area with the hexadecimal data abcd. When the job completes, the local data area is destroyed automatically by the operating system.

```
insert_job: i5job_lda
job_type: I5
machine: i5agent
i5_action: COMMAND
i5_name: IVP
i5_lda: x'abcd'
```

## i5\_lib Attribute—Specify the Name of a Library

The i5\_lib attribute specifies the name of the library that contains the Control Language (CL) program, the source file for the CL program, or the command that you want to run.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following formats:

```
i5_lib: library
i5_lib: /QSYS.LIB/library.LIB
```

#### *library*

Specifies the name of the library that contains the program, the source file for the program, or the command that you want to run. The value must be a valid i5/OS library name.

#### Notes:

- The entire value can be up to 46 characters; it cannot contain delimiters (such as spaces).
- Instead of using the i5\_lib attribute, you can specify the library name in the i5\_name attribute, or you can specify the library name using the i5\_curr\_lib or i5\_library\_list attribute.

- Do not use the i5\_lib attribute if the library name is already specified in the i5\_name attribute. If you specify the library name in both attributes, you will get a submission error. The job status will be JOB FAILURE.
- If you do not specify the library name in the i5\_name attribute or the i5\_lib attribute, the agent uses the job's library list.

#### Example: Specify the Library that Contains the CL Program to Run

This example runs a program named MFGDATA on an i5/OS system. The MFGDATA program is contained in the library named PRODJOBS.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: MFGDATA
i5_lib: PRODJOBS
```

## i5\_library\_list Attribute—Specify the Libraries for an i5/OS Job

The i5\_library\_list attribute specifies the names of the libraries that the job uses.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

```
i5_library_list: "library..."
```

#### *library*

Specifies the name of a library that you want to add to the library list.

**Limits:** Up to 10 characters; it cannot contain delimiters (such as spaces)

#### Notes:

- You can specify up to 25 libraries separated by spaces. The libraries are searched in the same order as they are listed.
- The entire value can be up to 274 characters.
- Enclose the entire list in quotation marks.

- The `i5_library_list` attribute specifies the initial user part of the library list that is used to search for operating system object names that do not have a library qualifier.
- The `i5_library_list` attribute overrides the library list specified in the job description for the job.

**[Example: Add a Library to the Library List of an i5/OS Job](#)**

This example runs a program named PAYJOB on an i5/OS system. The library named PAYROLL is added to the library list.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYJOB
i5_library_list: "PAYROLL"
```

**[Example: Add More Than One Library to the Library List of an i5/OS Job](#)**

This example runs a program named ACCTJOB on an i5/OS system. The libraries named PAYROLL and FINANCE are added to the library list.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: ACCTJOB
i5_library_list: "PAYROLL FINANCE"
```

## i5\_name Attribute—Specify a CL Program, Source File, or Command

The i5\_name attribute specifies the program, the source file for the program, or the command that you want to run on an i5/OS system.

**Note:** Use this attribute with the i5\_action attribute, which specifies whether the job runs a program or issues a command. The format of this attribute value depends on the action type specified in the i5\_action attribute.

### Supported Job Type

This attribute is required for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following formats:

- To specify a program:

```
i5_name: program
i5_name: library/program
i5_name: /QSYS.LIB/library.LIB/program.PGM
```

**Notes:**

- The entire value can be up to 60 characters; it cannot contain delimiters (such as spaces).
- To specify a program, the value of the i5\_action attribute must be RUN\_PROGRAM.

- To specify the source file for a program:

```
i5_name: file
i5_name: file(member)
i5_name: library/file(member)
i5_name: /QSYS.LIB/library.LIB/file.FILE/member.MBR
```

**Notes:**

- The entire value can be up to 60 characters; it cannot contain delimiters (such as spaces).
- To specify a source file, the value of the i5\_action attribute must be RUN\_PROG\_IN\_FILE.

- To specify a command:

*i5\_name: command*

*i5\_name: library/command*

*i5\_name: /QSYS.LIB/library.LIB/command.CMD*

**Notes:**

- The entire value can be up to 120 characters; enclose values that contain delimiters (such as spaces) in single quotation marks.
- To specify a command, the value of the *i5\_action* attribute must be COMMAND.

***program***

Specifies the name of the i5/OS program to run. The value must be a valid i5/OS program name.

***library***

Specifies the name of the library that contains the program, the source file for the program, or the command that you want to run. The value must be a valid i5/OS library name.

***file***

Specifies the file containing the CL source for the program. The value must be a valid i5/OS file name.

**Note:** If you specify the file name without a member name, \*FIRST is used by default.

***member***

Specifies the member in the file that contains the CL source for the program. The value must be a valid i5/OS member name.

***command***

Specifies the name of the i5/OS command to run. The value must be a valid i5/OS command name.

**Notes:**

- If the library name is already specified in the *i5\_name* attribute, do not use the *i5\_lib* attribute. If you specify the library name in both attributes, you will get a submission error. The job status will be JOB FAILURE.
- If you do not specify the library name in the *i5\_name* attribute or the *i5\_lib* attribute, the agent uses the job's library list.

### Example: Specify a Program to Run

This example runs a program named MFGDATA on an i5/OS system. The MFGDATA program is contained in the library named PRODJOBS.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: MFGDATA
i5_lib: PRODJOBS
```

### Example: Specify a File that Contains the Program to Run

This example runs a program that is contained in the BACKUP member of the CLSRC file. The library name is not specified in the i5\_name attribute or the i5\_lib\_attribute, so the agent uses the job's library list.

```
insert_job: i5job_file
job_type: I5
machine: i5agent
i5_action: RUN_PROG_IN_FILE
i5_name: CLSRC(BACKUP)
```

### Example: Specify a Command to Run

This example issues the DSPLIBL command to display the job's library list. When the job completes, you can view the library list in the job's spool file.

```
insert_job: i5job_displib
job_type: I5
machine: i5agent
i5_action: COMMAND
i5_name: DSPLIBL
i5_lib: *LIBL
i5_curr_lib: AGTLIB
i5_job_name: PAYJOBA
i5_job_desc: PRDLIB/PRDOPR
i5_job_queue: ABCSYS/PAYROLL
```

## i5\_others Attribute—Pass Keyword Parameters to SBMJOB

When CA Workload Automation AE submits a job to the i5/OS system, the job must pass through the SBMJOB command to execute. Keyword parameters are additional parameters that CA Workload Automation AE passes to the i5/OS SBMJOB command. The i5\_others attribute lets you specify any valid SBMJOB command keyword and value combination. You can also specify multiple combinations.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

`i5_others: keyword=value[,keyword=value...]`

#### **keyword=value**

Specifies any valid SBMJOB command keyword and value combination. The value corresponds to the SBMJOB command keyword (the value that would appear in brackets after the keyword in the SBMJOB command).

**Note:** Enclose values that contain spaces in single quotation marks.

#### **Notes:**

- You can specify up to 64 *keyword=value* combinations. Separate each combination with a comma.
- Each *keyword=value* combination can be up to 1024 characters.
- You can use multiple entries of i5\_others to define different sets of *keyword=value* combinations. For example, you can specify two i5\_others attributes as follows:

```
i5_others: PRTDEV=*J0BD
i5_others: OUTQ=*J0BD
```

The entire value of each i5\_others entry can be up to 4096 characters.

- CA Workload Automation AE does not validate the keyword and value combinations.
- The agent uses the user's default output queue if the owner attribute is specified in the job definition and the i5\_others attribute does not include the SBMJOB OUTQ() command keyword and value that defines the output queue for the job.

### Example: Specify the Printer and Output Queue for an i5/OS Job

This example runs a program named PAYJOB on an i5/OS system. The printer and output queue information is taken from the job definition.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYJOB
i5_others: PRTDEV=*JOBDEF,OUTQ=*JOBDEF
```

## i5\_params Attribute—Pass Positional Parameters

The i5\_params attribute defines the parameter values that you want to pass to the program at the time the program is invoked.

**Note:** To use this attribute, you must specify RUN\_PROGRAM or COMMAND for the i5\_action attribute.

### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

### Syntax

This attribute has the following format:

i5\_params: value[,value...]

**value[,value...]**

Specifies one or more parameter values to pass to the program. Separate each parameter with a comma.

**Limits:** Up to 64 parameters; each parameter can be up to 1024 characters

**Note:** To pass a parameter containing spaces, enclose it in quotation marks.

### Notes:

- You must specify each parameter in the order it is expected in the program.
- You can specify multiple i5\_params attributes in a job definition, as shown in the following example:

```
i5_params: parameter1, parameter2, parameter3
i5_params: parameter4, parameter5, parameter6
i5_params: parameter7, parameter8, parameter9
```

#### Example: Pass Multiple Parameters to an i5/OS Job

This example passes six parameters to an i5/OS program named PAYJOB. The parameter VALUE C is enclosed with double quotation marks because it contains a space.

```
insert_job: i5job_lib
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYJOB
i5_params: ABC 1 P "VALUE C" X r
```

## i5\_process\_priority Attribute—Specify the Process Priority of an i5/OS Job

The i5\_process\_priority attribute specifies the process priority of an i5/OS job. Process priority determines the order in which processes are scheduled on the processor. Depending on the priority level, process priority can speed up or slow down a process.

**Note:** If you do not specify the i5\_process\_priority attribute in your job definition, the job uses the normal priority (the default).

#### Supported Job Type

This attribute is optional for the [i5/OS job type](#) (see page 224).

#### Syntax

This attribute has the following format:

i5\_process\_priority: HIGH | ABOVE\_NORMAL | NORMAL | BELOW\_NORMAL | IDLE

#### HIGH

Indicates processes that must be executed immediately. These processes can use nearly all available CPU time.

#### ABOVE\_NORMAL

Indicates processes that have priority above the normal level but below the high level.

#### NORMAL

Indicates processes without special scheduling needs. This is the default.

#### BELLOW\_NORMAL

Indicates processes that have priority above the idle level but below the normal level.

#### IDLE

Indicates processes that will run only when the system is idle.

**Note:** For UNIX jobs running on an i5/OS system, you can increase the process priority above normal only if the agent is running under a profile that has \*JOBCTL authority. If the agent is not running under a profile that has \*JOBCTL authority and you set the process priority to a level above normal, the job runs with the normal process priority.

#### Example: Send a C Program's Return Code as the Job's Exit Code

This example sets the process priority for an i5/OS job to HIGH:

```
insert_job: i5_priority
job_type: I5
machine: i5agent
i5_action: RUN_PROGRAM
i5_name: PAYROLL
i5_process_priority: HIGH
```

## initial\_context\_factory Attribute—Specify an Initial Context Factory

The initial\_context\_factory attribute specifies the initial context factory to use when creating the initial context in a Entity Bean, Session Bean, JMS Publish, and JMS Subscribe job.

#### Supported Job Types

This attribute is required for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214)
- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

#### Syntax

This attribute has the following format:

```
initial_context_factory: factory
```

***factory***

Specifies the initial context factory to use when creating the initial context. The initial context is required within the Java Naming and Directory Interface (JNDI) framework. The initial context factory is supplied by a specific provider of the naming and directory service. The factory acquires an arbitrary initial context that the application can use to connect to the application server.

**Limits:** Up to 256 characters; case-sensitive

**Example:** weblogic.jndi.WLInitialContextFactory

**Example: Specify Initial Context Factory for a WebSphere Application Server**

Suppose that you want to invoke the reverse method on the CybEJBTestBean stateless session bean. The reverse method has one parameter, with type java.lang.String and value a23. The output from the reverse method is saved in the C:\Makapt15 file. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere application server and 2809 is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

## inside\_range Attribute—Specify Whether to Monitor for CPU Usage Within or Outside a Range

The inside\_range attribute specifies whether the job completes (or triggers if monitoring continuously) when the CPU or disk usage value is inside or outside the specified boundaries.

**Note:** If you do not specify the inside\_range attribute in your job definition, the job completes or triggers if the usage value is inside the lower and upper boundaries (the default).

### Supported Job Types

This attribute is optional for the following job types:

[CPU Monitoring \(OMCPU\) job type](#) (see page 206)

[Disk Monitoring \(OMD\) job type](#) (see page 213)

To use this attribute, you must specify WAIT or CONTINUOUS for the monitor\_mode attribute. This attribute does not apply when the monitor mode is NOW.

#### Syntax

This attribute has the following format:

inside\_range: TRUE | FALSE

#### TRUE

Specifies that the job completes (or triggers if monitoring continuously) when the value of CPU usage or disk usage is inside the range specified by the lower\_boundary and upper\_boundary attributes. This is the default.

#### **FALSE**

Specifies that the job completes (or triggers if monitoring continuously) when the value of CPU usage or disk usage is outside the range specified by the lower\_boundary and upper\_boundary attributes.

#### **Example: Monitor Available CPU**

This example continuously monitors the CPU available on the winagt computer. An alert is written to the scheduler log file each time the available CPU is less than 25 percent or greater than 75 percent, and the CPU usage changes by more than 10 percent. If the change in value is less than or equal to 10, the job does not register a change.

```
insert_job: omcpu_job
job_type: OMCPU
machine: winagt
cpu_usage: FREE
lower_boundary: 25
upper_boundary: 75
inside_range: FALSE
no_change: 10
monitor_mode: CONTINUOUS
```

#### **Example: Monitor Disk Space Usage Outside of a Range**

This example monitors the C drive on a Windows computer for used space. When the value of disk space used falls below 16 percent or exceeds 95 percent, the job completes.

```
insert_job: omd_win
job_type: OMD
machine: winagent
disk_drive: C
disk_format: PERCENT
lower_boundary: 16
upper_boundary: 95
inside_range: FALSE
disk_space: USED
```

## interactive Attribute—Specify Whether to Run a Command Job in Interactive Mode on Windows

The interactive attribute specifies whether to submit a Command job in interactive mode or in batch mode on Windows. Interactive mode lets users view and interact with jobs that invoke Windows Terminal Services or user interface processes. For example, you can define a job to open a GUI application, such as Notepad, on the Windows desktop.

**Note:** If you do not specify the interactive attribute in your job definition, the job runs in batch mode (the default).

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204) on Windows.

### Syntax

This attribute has the following format:

interactive: y | n

**y**

Runs the Command job in interactive mode on Windows.

**n**

Runs the Command job in batch (background) mode on Windows. This is the default.

### Notes:

- This attribute is ignored for Command jobs that run on UNIX.
- If the agent administrator sets the oscomponent.interactive parameter to true in the agentparm.txt file, the agent submits all Windows jobs in interactive mode, regardless of the interactive attribute. To use the interactive attribute in a job definition, the oscomponent.interactive parameter must be set to false.
- By default, the agent uses the default Windows shell explorer.exe. The agent administrator can specify an alternative Windows shell for interactive jobs by setting the oscomponent.shell parameter in the agentparm.txt file.

#### **Example: Run a Command Job in Interactive Mode on Windows**

This example runs a Command job in interactive mode on Windows. The job opens the config.txt file in the Windows notepad application on the Windows desktop.

```
insert_job: edit_file
job_type: CMD
machine: winagent
description: "Edit/review a configuration file"
command: notepad.exe "c:\run_info\config.txt"
interactive: y
```

## **invocation\_type Attribute—Specify the HTTP Method Type**

The invocation\_type attribute indicates the HTTP method type, which can be either GET or POST, in an HTTP job.

**Note:** If you do not specify the invocation\_type attribute in your job definition, the job uses the POST method type (the default).

### **Supported Job Type**

This attribute is optional for the [HTTP job type](#) (see page 222).

### **Syntax**

This attribute has the following format:

`invocation_type: GET | POST`

#### **GET**

Sends the URL over HTTP using the GET method. The GET method requests data and sends the data as part of the URL.

#### **POST**

Sends the URL over HTTP using the POST method. The POST method submits data and is the preferred method for sending lengthy form data. This is the default.

### Example: Use the GET Method to Perform a Google Search

Suppose that you want to define a job to perform a Google search and have the results returned to the job's spool file. You also want to refine your search results by specifying a filter. In this example, the job uses the HTTP GET method to perform the Google search on "ca workload automation". When the job runs, the job's spool file includes all matches that contain the filter AE.

```
insert_job: google
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://google.com/search"
j2ee_authentication_order: BASIC,DIGEST,NTLM
filter: .*AE.*
j2ee_parameter: q="ca workload automation"
```

## ip\_host Attribute—Specify the IP Address to Monitor

The ip\_host attribute specifies an IP address to monitor.

### Supported Job Type

This attribute is required for the [IP Monitoring \(OMIP\) job type](#) (see page 226).

### Syntax

This attribute has the following format:

ip\_host: *ip\_address*

***ip\_address***

Specifies the DNS name or IP address.

**Limits:** Up to 100 characters; case-sensitive

**Example:** 172.24.36.107 (IPv4) or 0:0:0:0:FFFF:192.168.00.00 (IPv6)

**Example: Monitor an IP Address Specified in Dotted Decimal Format**

This example monitors a device at IP address 172.31.255.255 and port 7510. When the job runs, it immediately checks if the device is running. If the device is running, the job completes successfully.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: 172.31.255.255
ip_port: 7510
monitor_mode: NOW
ip_status: RUNNING
```

## ip\_port Attribute—Specify the Port Number at the IP Address to Monitor

The ip\_port attribute specifies the port number at the IP address to monitor.

### Supported Job Type

This attribute is optional for the [IP Monitoring \(OMIP\) job type](#) (see page 226).

### Syntax

This attribute has the following format:

```
ip_port: ip_port
ip_port
```

Specifies the port number for the IP address being monitored. The agent attempts to connect to this port.

**Limits:** 0-65535

**Example:** 5800

**Note:** If you specify 0 or no value for the port, the agent uses ping for monitoring.

### Example: Monitor an IP Address and Port

This example monitors a device at IP address 172.31.255.255 and port 7510. When the job runs, it immediately checks if the device is running. If the device is running, the job completes successfully.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: 172.31.255.255
ip_port: 7510
monitor_mode: NOW
ip_status: RUNNING
```

## ip\_status Attribute—Specify the Status of the IP Address to Monitor

The ip\_status attribute specifies the status of the IP address to monitor.

**Note:** If you do not specify the ip\_status attribute in your job definition, the job monitors the IP address for a stopped status (the default).

### Supported Job Type

This attribute is optional for the [IP Monitoring \(OMIP\) job type](#) (see page 226).

### Syntax

This attribute has the following format:

ip\_status: RUNNING | STOPPED

#### RUNNING

Specifies that the job monitors the IP address for a running status.

#### STOPPED

Specifies that the job monitors the IP address for a stopped status. This is the default.

### Example: Monitor an IP Address for a Running Status

This example monitors a device at IP address 172.31.255.255 and port 7510. When the job runs, it immediately checks if the device is running. If the device is running, the job completes successfully.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: 172.31.255.255
ip_port: 7510
monitor_mode: NOW
ip_status: RUNNING
```

**Example: Monitor an IP Address for a Stopped Status**

This example monitors a device with DNS name myhost. When the device stops running, the job completes.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: myhost
ip_status: STOPPED
monitor_mode: WAIT
```

## j2ee\_authentication\_order Attribute—Specify Authentication Protocols

The j2ee\_authentication\_order attribute specifies a list of protocols to be used by a web server for authentication in an HTTP job.

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following formats:

```
j2ee_authentication_order: BASIC | DIGEST | NTLM
```

```
j2ee_authentication_order: {[BASIC]}
 {,[DIGEST]}
 {,[NTLM]}
```

#### BASIC

Indicates the BASIC protocol.

#### DIGEST

Indicates the DIGEST protocol.

#### NTLM

Indicates the NTLM protocol.

#### Notes:

- You can specify any of the following protocols in any order: BASIC, DIGEST, and NTLM. To specify more than one protocol, separate the protocols by commas.
- If you are connecting to an old web server that cannot negotiate authentication protocols, enter the following list in the specified order: BASIC,DIGEST,NTLM.

#### Example: Connect to a Web Server that Cannot Negotiate Authentication Protocols

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_conn\_domain Attribute—Specify a Domain for NTLM Connection Authentication

The j2ee\_conn\_domain attribute specifies the domain for connection authentication in an HTTP job. This attribute is required for NTLM authentication.

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

j2ee\_conn\_domain: *origin\_domain*

*origin\_domain*

Specifies the domain for NTLM connection authentication.

**Limits:** Up to 80 characters; case-sensitive

#### Example: Specify Connection Domain and Origin for NTLM Authentication

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_conn\_origin Attribute—Specify an Origin Host Name for NTLM Connection Authentication

The j2ee\_conn\_origin attribute specifies the origin host name for NTLM connection authentication in an HTTP job.

**Note:** If you do not specify the j2ee\_conn\_origin attribute in your job definition, the job uses the computer name where the agent is running (the default).

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

```
j2ee_conn_origin: origin_host
origin_host
```

Specifies the origin host name for NTLM connection authentication.

**Limits:** Up to 80 characters; case-sensitive

**Default:** The computer name where the agent is running

### Example: Specify Connection Domain and Origin for NTLM Authentication

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_no\_global\_proxy\_defaults Attribute—Specify Whether to Use Global Proxy Defaults

The j2ee\_no\_global\_proxy\_defaults attribute specifies whether the agent uses the global proxy defaults specified in the agentparm.txt file.

**Note:** If you do not specify the j2ee\_no\_global\_proxy\_defaults attribute in your job definition, the job does not use the global proxy configuration specified by the proxy parameters in the agentparm.txt file (the default).

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

j2ee\_no\_global\_proxy\_defaults: Y | N

**Y**

Ignores the global proxy configuration specified by the proxy parameters in the agentparm.txt file. Specify this option if you want to ignore the proxy defaults for this job only. For example, you might ignore the proxy defaults if the request is going to a server on the LAN that does not require the proxy. This is the default.

**N**

Uses the global proxy configuration specified by the proxy parameters in the agentparm.txt file.

#### Example: Override Global Proxy Defaults in Agent Parameter File

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_parameter Attribute—Specify Input Parameters

The j2ee\_parameter attribute specifies input parameters in a job.

### Supported Job Types

This attribute is required for the [JMS Publish \(JMSPUB\) job type](#) (see page 228).

This attribute is optional for the following job types:

- [HTTP](#) (see page 222)
- [POJO](#) (see page 252)
- [RMI \(JAVARMI\)](#) (see page 254)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

### Syntax

This attribute has the following format:

```
j2ee_parameter: type=value | payload_job=job_name[, type=value |
payload_job=job_name...]
```

**type=value**

Specifies the parameter type and value. In most job types, it specifies the Java class and String value of the parameter. In an HTTP job, it specifies the variable name and value that the agent passes to the program the job invokes.

**Limits:** Up to 1024 characters; case-sensitive

**Examples:** String="5:00PM", studentName=SMITH

**Note:** If the package of the Java class is not specified, java.lang is assumed.

**payload\_job=job\_name**

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

**Note:** This parameter does not apply to HTTP jobs.

**Notes:**

- You can define up to 64 parameters using one j2ee\_parameter attribute or multiple entries of j2ee\_parameter.
- In each j2ee\_parameter entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. Specify the parameters in the same order as they appear in the method. The job fails if the parameters are listed in the wrong order.

**JMS Publish Notes:**

- To construct a text message, you require a single parameter (String). To construct an object message, the number of parameters that you require depends on the object's constructor. For example, an Integer object requires a single constructor (String). Some objects require no parameters or multiple parameters for construction.
- Leading and trailing spaces are not supported in JMS Publish messages.

**Example: Define an RMI Job to Start a Remote Server Immediately**

Suppose that you want to invoke a method that starts a remote server using remote object activation. You want the server to start immediately.

```
insert_job: start
job_type: JAVARMI
machine: appagent
remote_name: "rmi://remotehost/Test"
method_name: startserver
j2ee_parameter: String="now"
```

#### Example: Define an HTTP Job to Perform a Google Search

Suppose that you want to define a job to perform a Google search and have the results returned to the job's spool file. You also want to refine your search results by specifying a filter. In this example, the job uses the HTTP GET method to perform the Google search on "ca workload automation". When the job runs, the job's spool file includes all matches that contain the filter AE.

```
insert_job: google
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://google.com/search"
j2ee_authentication_order: BASIC,DIGEST,NTLM
filter: .*AE.*
j2ee_parameter: q="ca workload automation"
```

## j2ee\_proxy\_domain Attribute—Specify a Domain for Proxy Authentication

The j2ee\_proxy\_domain attribute specifies the domain for proxy authentication in an HTTP job. This attribute is required for NTLM authentication.

**Note:** If you do not specify the j2ee\_proxy\_domain attribute in your job definition, the job uses the proxy domain set in the http.proxyDomain parameter on the agent.

#### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

#### Syntax

This attribute has the following format:

*j2ee\_proxy\_domain: proxy\_domain*

#### *proxy\_domain*

Specifies the domain for proxy authentication.

**Limits:** Up to 80 characters; case-sensitive

#### Notes:

- Your agent administrator can specify a default proxy domain for all HTTP jobs by setting the http.proxyDomain parameter in the agent's agentparm.txt file.
- The j2ee\_proxy\_domain attribute overrides the default proxy domain specified in the agent's agentparm.txt.

### Example: Override Global Proxy Defaults in Agent Parameter File

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_proxy\_host Attribute—Specify a Proxy Host

The j2ee\_proxy\_host attribute specifies the proxy host name to use for the request in an HTTP job.

**Note:** If you do not specify the j2ee\_proxy\_host attribute in your job definition, the job uses the proxy host set in the http.proxyHost parameter on the agent.

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

j2ee\_proxy\_host: *proxy\_host*

#### *proxy\_host*

Specifies the proxy host name to use for the request.

**Limits:** Up to 256 characters; case-sensitive

**Notes:**

- Your agent administrator can specify a default proxy host name for all HTTP jobs by setting the http.proxyHost parameter in the agent's agentparm.txt file.
- The j2ee\_proxy\_host attribute overrides the default proxy host name specified in the agent's agentparm.txt.

**Example: Override Global Proxy Defaults in Agent Parameter File**

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_proxy\_origin\_host Attribute—Specify an Origin Host Name for Proxy Authentication

The j2ee\_proxy\_origin\_host attribute specifies the origin host name for proxy authentication in an HTTP job. This attribute is used for NTLM authentication.

**Note:** If you do not specify the j2ee\_proxy\_origin\_host attribute in your job definition, the job uses the computer name where the agent is running (the default).

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

```
j2ee_proxy_origin_host: proxy_origin
proxy_origin
```

Specifies the origin host name for proxy authentication.

**Limits:** Up to 256 characters; case-sensitive

**Default:** The computer name where the agent is running

### Example: Override Global Proxy Defaults in Agent Parameter File

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_proxy\_port Attribute—Specify a Proxy Port

The j2ee\_proxy\_port attribute specifies the proxy port to use for the request in an HTTP job.

**Note:** If you do not specify the j2ee\_proxy\_port attribute in your job definition, the job uses port 80 (the default).

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

```
j2ee_proxy_port: proxy_port
proxy_port
Specifies the proxy port to use for the request.
Default: 80
Limits: 0-65535
```

### Example: Override Global Proxy Defaults in Agent Parameter File

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## j2ee\_proxy\_user Attribute—Specify a User for Proxy Authentication

The j2ee\_proxy\_user attribute specifies the user name required for proxy authentication.

**Note:** If you do not specify the j2ee\_proxy\_user attribute in your job definition, the job uses the proxy user set in the http.proxyUser parameter on the agent.

### Supported Job Type

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

```
j2ee_proxy_user: proxy_user
proxy_user
```

Specifies the user name required for proxy authentication.

**Limits:** Up to 80 characters; case-sensitive; it cannot contain delimiters (such as spaces)

### Notes:

- Your agent administrator can specify a default proxy user name for all HTTP jobs by setting the http.proxyUser parameter in the agent's agentparm.txt file.
- The j2ee\_proxy\_user attribute overrides the default proxy user name specified in the agent's agentparm.txt.

#### **Example: Override Global Proxy Defaults in Agent Parameter File**

This example performs an HTTP query using the HTTP GET method. The output of the invocation is returned in the job's spool file. In this example, the job specifies the connection domain and origin for NTLM authentication, overrides the global proxy defaults specified in the agentparm.txt file, and specifies the BASIC, DIGEST, and NTLM protocols for web server authentication.

```
insert_job: HTTP.CON_USER
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://host.example.com/protected"
j2ee_conn_origin: host.example.com
j2ee_conn_domain: windows_domain
j2ee_no_global_proxy_defaults: Y
j2ee_authentication_order: BASIC,DIGEST,NTLM
j2ee_proxy_domain: "http://host.domain.proxy"
j2ee_proxy_host: proxy.example.com
j2ee_proxy_origin_host: "http://host.origin.proxy"
j2ee_proxy_port: 90
j2ee_proxy_user: user01
```

## **j2ee\_user Attribute—Specify a JNDI User Name**

The j2ee\_user attribute specifies the JNDI user name in an Entity Bean, Session Bean, JMS Publish, and JMS Subscribe job. This user name refers to the application server within the JNDI framework. If specified, this authentication information is supplied when creating the initial context.

### **Supported Job Types**

This attribute is optional for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214)
- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

### **Syntax**

This attribute has the following format:

```
j2ee_user: user_name
```

***user\_name***

Specifies the JNDI user ID to be used to gain access to the connection factory.

**Limits:** Up to 145 characters; case-sensitive; it cannot contain delimiters (such as spaces)

**Example: Specify a JNDI User ID in a JMS Publish Job**

This example publishes the message "this is my message" to the queue named Queue. The Java class of the message is String. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere MQ server and 2809 is the ORB port. The job uses the cyberuser JNDI user name to gain access to the connection factory named ConnectionFactory.

```
insert_job: publish
job_type: JMSPUB
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
connection_factory: ConnectionFactory
destination_name: Queue
use_topic: FALSE
message_class: String
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="this is my message"
```

**Note:** The agent does not support JMS messaging on IBM WebSphere. If you have IBM WebSphere MQ, your agent administrator can set up the agent plug-in to run JMS Publish and JMS Subscribe for JMS queues. JMS topics are not supported on IBM WebSphere MQ.

## jcl\_library Attribute—Specify a JCL Library

The jcl\_library attribute specifies the JCL library that contains the JCL for your job.

### Supported Job Type

This attribute is required for the [z/OS Regular \(ZOS\) job type](#) (see page 285).

### Syntax

This attribute has the following format:

`jcl_library: library`

#### *library*

Specifies the Job Control Language (JCL) library name. The JCL library or JCLLIB contains the JCL for the z/OS job. The JCLLIB is a z/OS data set name. Data set names typically have up to four positions. Each position can be up to eight characters. Separate the positions using periods. The JCL library can be any one of the following:

- A valid z/OS data set name. You can specify a partitioned data set (PDS) or a sequential data set.
- A valid z/OS data set name, prefixed with pan- to indicate Panvalet input
- A valid z/OS data set name, prefixed with lib- to indicate Librarian input

**Limits:** Up to 44 characters

**Note:** If you specify a partitioned data set, you must also specify the jcl\_member attribute in your job definition.

### Example: Specify a JCL Library

This example submits the JCL in member CYBDL01A in the CYBDL01.JCLLIB library.

```
insert_job: CYBDL01A
job_type: ZOS
machine: ZOS1
jcl_library: CYBDL01.JCLLIB
jcl_member: CYBDL01A
```

## jcl\_member Attribute—Specify a JCL Member

The jcl\_member attribute specifies the JCL member that contains the JCL for your job.

### Supported Job Type

This attribute is required for the [z/OS Regular \(ZOS\) job type](#) (see page 285) if a partitioned data set (PDS) is specified for the jcl\_library attribute.

### Syntax

This attribute has the following format:

jcl\_member: *member*

***member***

Specifies the JCL member that contains the JCL for your job.

**Limits:** Up to 8 characters

### Example: Specify a JCL Member

This example submits the JCL in member CYBDL01A in the CYBDL01.JCLLIB library.

```
insert_job: CYBDL01A
job_type: ZOS
machine: ZOS1
jcl_library: CYBDL01.JCLLIB
jcl_member: CYBDL01A
```

## jmx\_parameter Attribute—Specify JMX Parameters

The jmx\_parameter attribute specifies the new value that the attribute should be set to in a JMX-MBean Attribute Set job, the CREATE parameter list in a JMX-MBean Create Instance job, and the parameter list in a JMX-MBean Operation job.

### Supported Job Types

This attribute is optional for the following job types:

- [JMX-MBean Attribute Set \(JMXMLS\)](#) (see page 233)
- [JMX-MBean Create Instance \(JMXMLC\)](#) (see page 235)
- [JMX-MBean Operation \(JMXMLOP\)](#) (see page 237)

### Syntax

This attribute has the following format:

```
jmx_parameter: type=value | payload_job=job_name[, type=value |
payload_job=job_name...]
```

#### **type=value**

Specifies the Java class and String value for the parameter.

**Limits:** Up to 1024 characters; case-sensitive

**Examples:** String="5:00PM", String[2]="winnt1,winnt2", double=2000

**Note:** If the package of the Java class is not specified, java.lang is assumed.

#### **payload\_job=job\_name**

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

### Notes:

- You can define only a single parameter in a JMX-MBean Attribute Set (JMXMLS) job.
- You can define up to 64 parameters in a JMX-MBean Create Instance (JMXMLC) job or a JMX-MBean Operation (JMXMLOP) job. Specify the parameters using one jmx\_parameter attribute or multiple entries of jmx\_parameter.
- In each jmx\_parameter entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. Specify the parameters in the attribute in the same order as they appear in the parameter list. The job fails if the parameters are listed in the wrong order.

**Example: Change the Value of an MBean Attribute Using a Serialized Java Object**

Suppose that you want to set the value of the State attribute of the cdc.jmx.SimpleDynamic MBean to the serialized Java object returned by a JMX-MBean Attribute Set job named size.

```
insert_job: change
job_type: JMXMLAS
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic"
mbean_attr: State
jmx_parameter: payload_job=size
condition: success(size)
```

**Example: Invoke an Operation on an MBean to Reset the Value of a Parameter**

Suppose that you want to invoke the resetmem operation on the config MBean to reset the value of the memory parameter to 50.

```
insert_job: reset
job_type: JMXMLOP
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_operation: resetmem
jmx_parameter: Integer=50
```

## jmx\_user Attribute—Specify a JMX User Name

The jmx\_user attribute specifies the user name in a JMX job.

### Supported Job Types

This attribute is optional for the following job types:

- [JMX-MBean Attribute Get \(JMXMLAG\)](#) (see page 232)
- [JMX-MBean Attribute Set \(JMXMLAS\)](#) (see page 233)
- [JMX-MBean Create Instance \(JMXMLC\)](#) (see page 235)
- [JMX-MBean Operation \(JMXMLOP\)](#) (see page 237)
- [JMX-MBean Remove Instance \(JMXMLREM\)](#) (see page 239)
- [JMX-MBean Subscribe \(JMXMLSUB\)](#) (see page 240)

### Syntax

This attribute has the following format:

`jmx_user: user_name`

***user\_name***

Specifies the user name.

**Limits:** Up to 145 characters

## job\_class Attribute—Assign a Job Class to a Job

The job\_class attribute assigns a job class value to a job. When this attribute is specified, the agent interprets the job class against the job classes defined in the `agentparm.txt` file. Each defined job class has a specified number of initiators, which allows for manual load balancing against system resources.

### Supported Job Types

This attribute is optional and applies to all job types *except* the z/OS job types (ZOS, ZOSM, and ZOSDST).

### Syntax

This attribute has the following format:

`job_class: class`

***class***

Specifies the job class that this job runs under. The agent maintains a list of job classes and the number of initiators assigned to each job class. A job class with more initiators can process more jobs more quickly. For higher-priority jobs, assign a job class that contains more initiators.

**Limits:** Up to 32 characters; it cannot contain delimiters (such as spaces)

**Notes:**

- The specified class must be defined in the agentparm.txt file. If the class name is unavailable on the target machine, the job submission will fail.
- If all the initiators in a specified class are being used, the job does not run. The job waits until the class has an initiator available.
- The agent administrator must specify the number of initiators for job classes in the agentparm.txt file. The following example specifies three job classes and a default job class:

```
initiators.class_1=Default,1000
initiators.class_2=JOBCLASS1,20
initiators.class_3=JOBCLASS2,500
initiators.class_4=FILEM,1000
```
- The agent administrator can set the initiators.afmjobclassmap\_N parameter to automatically assign a job class based on the combination of verbs and subverbs that appear in a message. For more information about the parameter, see the *Implementation Guide* for CA WA Agent for UNIX, Linux, Windows, or i5/OS.
- The job\_class attribute overrides the default job class specified in the agent's agentparm.txt.

**Example: Specify a Job Class in a Command Job on UNIX**

This example runs a UNIX Command job under job class JOBCLASS2.

```
insert_job: unix_cmd
job_type: CMD
machine: unixagent
command: /export/home/payroll/data
job_class: JOBCLASS2
```

## job\_load Attribute—Define a Job's Relative Processing Load

The job\_load attribute defines the relative amount of processing power the job you are defining consumes. The range of possible values is arbitrary and user-defined. You can use the max\_load attribute to assign a machine a “maximum job load” (that is, a measure of CPU load that is desirable to place on a machine at any given time). Then, you can use the job\_load attribute to assign the job a load value that indicates the relative amount of the machine's load that the job should consume. This scheme allows you to control machine loading and to prevent a machine from being overloaded.

When a job is ready to run on a designated machine but the current load on that machine is too large to accept the new job's load, CA Workload Automation AE queues the job for that machine so that it runs when sufficient resources are available. However, for this scheme to function properly, you must specify a job load for each job to run on a controlled machine; otherwise, a job could start on a machine without the machine showing the additional load.

When you force a job to start, for example by using the sendevent command to issue a FORCE\_STARTJOB event, the job runs even if its load exceeds the machine's max\_load value.

The default job priority is 0, which means that the job should run immediately and ignore any available load units. Therefore, whenever you set job\_load for a job, you should also set the priority attribute for the job to 1 or higher to ensure that the job\_load setting is used.

**Note:** You cannot use the priority or job\_load attribute if you specify a user-defined load balancing script in the machine attribute.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`job_load: load_units`

#### *load\_units*

Defines the relative load of the job. This number can be any value in the user-defined range of possible values, which are arbitrary. Define a value that has some relationship to the value defined in the max\_load attribute.

**Default:** 0 (unrestricted)

**Limits:** Up to 10 digits

### Example: Define the Relative Processing Load for a Job

This example sets the load for a job that typically uses 10% of the CPU, with a range of possible load values from 1-100.

```
job_load: 10
```

## job\_terminator Attribute—Kill a Job if Its Box Fails

The job\_terminator attribute specifies whether to use a KILLJOB event to terminate the job if its containing box job completes with a FAILURE or TERMINATED status.

Use this attribute with the box\_terminator attribute to control how nested jobs behave when a job fails.

### Supported Job Types

This attribute only applies to jobs that are in a box. This attribute is optional for the following job types:

- [Box](#) (see page 201)
- [Command \(CMD\)](#) (see page 204)
- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Database Monitor \(DBMON\)](#) (see page 207)
- [Database Stored Procedure \(DBPROC\)](#) (see page 209)
- [Database Trigger \(DBTRIG\)](#) (see page 211)
- [Disk Monitoring \(OMD\)](#) (see page 213)
- [File Trigger \(FT\)](#) (see page 217)
- [File Watcher \(FW\)](#) (see page 219)
- [IP Monitoring \(OMIP\)](#) (see page 226)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [JMX-MBean Subscribe \(JMXSUB\)](#) (see page 240)
- [Oracle Applications Copy Single Request \(OACOPY\)](#) (see page 242)
- [Oracle Applications Request Set \(OASET\)](#) (see page 244)

- [Oracle Applications Single Request \(OASG\)](#) (see page 246)
- [PeopleSoft \(PS\)](#) (see page 249)
- [Process Monitoring \(OMP\)](#) (see page 253)
- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP BW InfoPackage \(SAPBWIP\)](#) (see page 256)
- [SAP BW Process Chain \(SAPBWPC\)](#) (see page 258)
- [SAP Data Archiving \(SAPDA\)](#) (see page 259)
- [SAP Event Monitor \(SAPEVT\)](#) (see page 261)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP Process Monitor \(SAPPM\)](#) (see page 264)
- [SAP R/3 \(SAP\)](#) (see page 266)
- [SQL](#) (see page 272)
- [Text File Reading and Monitoring \(OMTF\)](#) (see page 274)
- [Windows Event Log Monitoring \(OMEL\)](#) (see page 279)
- [Windows Service Monitoring \(OMS\)](#) (see page 281)
- [z/OS Manual \(ZOSM\)](#) (see page 284)
- [z/OS Regular \(ZOS\)](#) (see page 285)

## Syntax

This attribute has the following format:

job\_terminator: Y | N

**Y**

Terminates the job if the containing box fails or terminates.

**Note:** You can specify 1 instead of Y.

**N**

Does not terminate the job if the containing box fails or terminates. This is the default.

**Note:** You can specify 0 instead of N.

**Windows Note:** Windows does not support the concept of process groups. When you issue a KILLJOB event for a job that runs an executable (\*.exe), KILLJOB kills the process specified in the command definition. When you issue a KILLJOB event for a job that runs something other than an \*.exe (for example, \*.bat, \*.cmd, or \*.com), KILLJOB terminates only the CMD.EXE process that CA Workload Automation AE used to launch the job. The Job Status is set according to the return code of the killed CMD.EXE process and can be one of the following: SUCCESS, FAILURE, or TERMINATED. Processes launched by user applications or batch (\*.bat) files are not killed.

### Example: Terminate a Job on Box Failure

This example terminates the jobterm1 job if its containing box, unix\_box, fails or terminates.

```
insert_job: jobterm1
job_type: C
box_name: unix_box
command: sleep 120
machine: unixagent
job_terminator: y
```

## job\_type Attribute—Specify Job Type

The job\_type attribute specifies a job's type.

**Note:** If you do not specify the job\_type attribute in your job definition, the job type is set to CMD (the default).

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`job_type: type`

***type***

Specifies the type of the job that you are defining.

You can specify one of the following values:

- BOX
- CMD—Command
- DBMON—Database Monitor
- DBPROC—Database Stored Procedure
- DBTRIG—Database Trigger
- ENTYBEAN—Entity Bean
- FT—File Trigger
- FTP—File Transfer Protocol
- HTTP—HTTP
- I5—i5/OS
- JAVARMI—Java Remote Method Invocation
- JMSPUB—JMS Publish
- JMSSUB—JMS Subscribe
- JMXMAG—JMX-MBean Attribute Get
- JMXMAS—JMX-MBean Attribute Set
- JMXMC—JMX-MBean Create Instance
- JMXMOP—JMX-MBean Operation
- JMXMREM—JMX-MBean Remove Instance
- JMXSUB—JMX-MBean Subscribe

- OACOPY—Oracle E-Business Suite Copy Single Request
- OASET—Oracle E-Business Suite Request Set
- OASG—Oracle E-Business Suite Single Request
- OMCPU—CPU Monitoring
- OMD—Disk Monitoring
- OMEL—Windows Event Log Monitoring Jobs
- OMIP—IP Monitoring
- OMP—Process Monitoring
- OMS—Windows Service Monitoring
- OMTF—Text File Reading and Monitoring
- POJO—Plain Old Java Object
- PS—PeopleSoft
- SAPBDC—SAP Batch Input Session
- SAPBWIP—SAP BW InfoPackage
- SAPBWPC—SAP BW Process Chain
- SAPDA—SAP Data Archiving
- SAPEVT—SAP Event Monitor
- SAPJC—SAP Job Copy
- SAPPMM—SAP Process Monitor
- SAP—SAP R/3
- SCP—Secure Copy
- SESSBEAN—Session Bean
- SQL—Structured Query Language Command
- WBSVC—Web Service
- ZOS—z/OS Regular
- ZOSDST—z/OS Data Set Trigger
- ZOSM—z/OS Manual

**Default:** CMD

**Notes:**

- For backward compatibility, the following values are supported:
  - c (in place of CMD)
  - f (in place of FW)
  - b (in place of BOX)
- In CA Workload Automation AE r4.x, it was acceptable to enter, for example, "file" or "foo" for the file watcher job type, because only the first character of the input string was checked. However, in CA Workload Automation AE r11, the syntax checking no longer allows this. You must enter "b" or "box" for box jobs, "c" or "cmd" for command jobs, or "f" or "fw" for file watcher jobs.

**Example: Specify a CMD Job**

This example defines a Command job to run a script on a UNIX computer named unixprod.

```
insert_job: cmd_job1
job_type: CMD
machine: unixprod
command: "/usr/scripts/calcproc.sh"
```

## lower\_boundary Attribute—Define the Start of the Range to be Searched (OMTF Jobs)

The lower\_boundary attribute defines the start of the range to be searched in a text file.

This attribute is used with the text\_file\_mode attribute, which specifies the search mode (line, regular expression, or date and time).

### Supported Job Type

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### Syntax

This attribute has the following format:

```
lower_boundary: start_range
start_range
```

Defines the start of the range to be searched. The format of this value depends on the text file search mode. The formatting options are as follows:

- Numeric—Specify a numeric value if the search mode is LINE.
- Regular expression—Specify a regular expression if the search mode is REGEX.
- Date and time—Specify date and time values if the search mode is DATETIME. Define the date and time using the format specified by the time format.

**Limits:** Up to 256 characters; case-sensitive

### Notes:

- If you do not specify the text\_file\_mode attribute in the job definition, the search mode is LINE by default.
- If you do not specify the lower\_boundary attribute in the job definition, the job starts searching from the first line of the text file.
- When the search mode is REGEX, you can specify a regular expression in the lower\_boundary attribute. To compose a regular expression, follow the rules for Java class java.util.regex.Pattern. You can find these rules using a Google search for java pattern.

- You can use escape sequences in the regular expression. The following characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\).
  - \—back slash
  - |—vertical bar
  - (—opening parenthesis
  - )—closing parenthesis
  - [—opening bracket
  - {—opening brace
  - ^—caret (circumflex)
  - \$—dollar sign
  - \*—asterisk
  - +—plus sign
  - ?—question mark
  - .—period

For example, to match the characters \*.\* literally, specify \\*\.\\* in your regular expression. The backward slashes escape the characters' special meanings.

#### [Example: Define the Start of a Range When the Search Mode is LINE](#)

This example searches the c:\ca\log file in line mode. The job starts searching the content from line 143 of the file. The upper boundary is not defined, so the job searches to the last line of the file. The job completes successfully if the ERROR MESSAGE string is found.

```
insert_job: omtf_line
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 143
monitor_mode: NOW
```

#### Example: Define the Start of a Range When the Search Mode is REGEX

This example searches the c:\ca\log file in regular expression mode. The lower boundary is not defined, so the job searches the content from the first line of the file to the upper boundary (a line that contains the word service). The job completes successfully if the ARCHIVE string is found.

```
insert_job: omtf_regex
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ARCHIVE
text_file_mode: REGEX
upper_boundary: service
monitor_mode: NOW
```

#### Example: Define the Start of a Range When the Search Mode is DATETIME

This example searches the /export/home/logs/transmitter.log file in date and time mode. The job searches the content between May 20, 2010 at midnight and May 27, 2010 at 11:59 p.m. The date and time values are defined using the format specified in the time\_format attribute. The job completes successfully if the transmitted string is found.

```
insert_job: omtf_timedate
job_type: OMTF
machine: monagt
text_file_name: /export/home/logs/transmitter.log
text_file_filter: transmitted
text_file_mode: DATETIME
lower_boundary: "Thu May 20 00:00:00.000 EDT 2010"
upper_boundary: "Thu May 27 23:59:59.999 EDT 2010"
time_format: "EEE MMM dd HH:mm:ss.SSS zzz yyyy"
time_position: 12
monitor_mode: NOW
```

## lower\_boundary Attribute—Define the Minimum CPU or Disk Usage to Monitor (OMCPU and OMD Jobs)

The lower\_boundary attribute defines the minimum CPU or disk usage to monitor for.

### Supported Job Types

This attribute is required for the following job types if WAIT or CONTINUOUS is specified for the monitor\_mode attribute:

- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Disk Monitoring \(OMD\)](#) (see page 213)

This attribute does not apply when the monitor mode is NOW.

### Syntax

This attribute has the following format:

- To specify the lower boundary for CPU usage:

`lower_boundary: min_percent`

***min\_percent***

Defines the minimum amount of CPU usage to monitor for in percent.

**Default:** 0

**Limits:** 0-100

**Note:** If you specify the lower\_boundary attribute without the upper\_boundary attribute, the monitoring takes place between the lower boundary and 100 percent.

- To specify the lower boundary for disk usage:

`lower_boundary: min_value`

***min\_value***

Defines the minimum amount of disk usage to monitor for. The `disk_format` attribute specifies the unit for this value.

**Limits:**

- 0-100 (This limit applies when the disk format is PERCENT)
- Up to 20 characters (This limit applies for all other disk formats.)

**Example:** 35000000

**Note:** If the disk format is PERCENT, the lower boundary must be less than the upper boundary.

**Example: Specify a Lower Boundary of 25 Percent for Monitoring Available CPU**

This example continuously monitors the CPU available on the winagt computer. An alert is written to the scheduler log file each time the available CPU is less than 25 percent or greater than 75 percent, and the CPU usage changes by more than 10 percent. If the change in value is less than or equal to 10, the job does not register a change.

```
insert_job: omcpu_job
job_type: OMCPU
machine: winagt
cpu_usage: FREE
lower_boundary: 25
upper_boundary: 75
inside_range: FALSE
no_change: 10
monitor_mode: CONTINUOUS
```

**Example: Specify a Lower Boundary of 8 GB for Monitoring Available Disk Space**

This example monitors /export/home for available space. The job completes when the available space is between 8 and 10 gigabytes (GB), and the available space changes by more than 15 percent. If the change in value is less than or equal to 15, the job does not register a change.

```
insert_job: omd_unix
job_type: OMD
machine: unixagt
disk_drive: /export/home
disk_space: FREE
disk_format: GB
lower_boundary: 8
upper_boundary: 10
inside_range: TRUE
no_change: 15
```

## machine Attribute—Define the Client Where a Job Runs

The machine attribute defines how agent selection for the job you are defining will be determined. The owner of the job must have permission to access this machine and to execute the specified command on this machine. The machine must already be defined to CA Workload Automation AE (using the `insert_machine` subcommand). If the machine was not previously defined, the product will not create the job.

You cannot use the priority or `job_load` attribute if you specify a script for making the selection in the machine attribute.

### Supported Job Types

This attribute is required for all job types. This attribute does not apply to Box jobs.

### Syntax

This attribute has the following format:

```
machine: {machine_name [, machine_name]...} | `machine_chooser`}
```

#### *machine\_name*

Defines the name of a pre-defined machine object.

**Limits:** Up to 80 alphanumeric characters

**Note:** If you specify `machine: localhost`, the scheduler tries to resolve the `localhost` value when it runs the job. By default, the `localhost` value is resolved to the name of the machine where the scheduler was started. If the local machine definition setting is defined to a value other than “localhost”, the `localhost` value is resolved to the name of the specified real machine. For more information about the local machine definition setting, see the *Administration Guide* (UNIX) or the *Online Help* for the CA Workload Automation AE Administrator (Windows). For more information about how the `localhost` value is resolved, see the *User Guide*.

***machine\_chooser***

Defines the name of a program or user-written batch file for the scheduler to execute at run time to determine which machine to use. The scheduler runs this program, writes the name of the machine to its log file, and substitutes this output as the machine name. You must enclose the fully qualified program or batch file name in grave accent marks (`).

**Note:** When specifying drive letters for Windows in job definitions, you must escape the colon with backslashes. For example, 'C:\tmp' is valid; 'C:\tmp' is not.

**Example: Run a Job on Either of Two Machines**

This example specifies that the job should run on either the machine named prod or the machine named test:

```
machine: prod, test
```

**Example: Run a Batch File to Determine the Client on Which to Run a Job (Windows)**

This example runs the batch file C:\MYSTUFF\MYCHOSER.BAT at job run time to determine which machine to use:

```
machine: `C:\\MYSTUFF\\MYCHOSER.BAT`
```

**Example: Run a Script to Determine the Client on Which to Run a Job (UNIX)**

This example runs the script /usr/local/bin/my-machine-chooser at job run time to determine which machine to use:

```
machine: `/usr/local/bin/my-machine-chooser`
```

## max\_exit\_success Attribute—Specify Maximum Exit Code for Success

The max\_exit\_success attribute specifies the maximum exit code the job can finish with and still be considered successful. An exit code equal to or less than the specified value is considered a success. CA Workload Automation AE uses this attribute when a command can exit with more than one exit code, indicating either “degrees of success” or other conditions that cannot indicate a failure. This attribute is useful when defining complex branching logic based on real-time processing.

**Note:** If you do not specify the max\_exit\_success attribute in your job definition, the scheduling manager interprets only an exit code of zero (0) as job success (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [Command \(CMD\)](#) (see page 204)
- [i5/OS \(I5\)](#) (see page 224)

### Syntax

This attribute has the following format:

```
max_exit_success: exit_code
exit_code
```

Defines the maximum exit code that the job can exit with and be considered a success.

**Default:** 0

**Limits:** 0-119

### Example: Specify the Maximum Exit Code for Success

This example runs the backup script on a UNIX computer. The job is considered successful if it completes with an exit code of 2 or less.

```
insert_job: unix_script
job_type: CMD
machine: unixagent
command: /usr/common/backup
max_exit_success: 2
```

## max\_run\_alarm Attribute—Define the Maximum Run Time for a Job

The max\_run\_alarm attribute specifies the maximum run time (in minutes) that a job should require to finish normally. This “reasonability” test can catch errors, such as when a job is stuck in a loop or waiting on a system event that never occurs. If the job runs longer than the specified time, CA Workload Automation AE generates an alarm to alert the operator to take corrective action.

**Note:** You can use the term\_run\_time attribute to automatically terminate a job that has run for too long. If you do not define the term\_run\_time attribute for a job, it continues running until it completes or you manually interrupt it.

### Supported Job Types

This attribute is optional for all job types.

We recommend that you define the max\_run\_alarm attribute for file watcher jobs to keep them from running indefinitely. For example, defining the max\_run\_alarm attribute for a file watcher job would address situations in which a communication link is down and preventing the arrival of a file.

### Syntax

This attribute has the following format:

`max_run_alarm: mins`  
***mins***

Defines the maximum number of minutes the job should ever require to finish normally. This value can be any integer.

**Default:** No maximum set.

**Limits:** 0-2,147,483,647

### Example: Define the Maximum Run Time for a Job

This example specifies that the product should generate an alarm if the job runs for more than two hours:

`max_run_alarm: 120`

## mbean\_attr Attribute—Specify the MBean Attribute to Query or Set

The mbean\_attr attribute specifies the name of the MBean attribute you want to query or set in a JMX-MBean Attribute job. You specify the mbean\_attr attribute to query a JMX server for the value of an MBean attribute or to change the value of an MBean attribute on a JMX server. You can change the value of an MBean attribute using a set value for the attribute or using a serialized Java object passed by another job.

### Supported Job Types

This attribute is required for the following job types:

- [JMX-MBean Attribute Get \(JMXMLG\)](#) (see page 232)
- [JMX-MBean Attribute Set \(JMXMLS\)](#) (see page 233)

### Syntax

This attribute has the following format:

`mbean_attr: attribute`

#### *attribute*

Specifies the name of the MBean attribute that you want to query or set.

**Limits:** Up to 256 characters; case-sensitive

### Example: Query a JMX Server for the Value of an MBean Attribute

Suppose that you want to know the value of the cachesize attribute for the Config MBean. The URL for the JMX server is service:jmx:rmi:///jndi/rmi://localhost:9999/server, where localhost is the host name and 9999 is the port number.

```
insert_job: query
job_type: JMXMLG
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_attr: cachesize
```

#### Example: Change the Value of an MBean Attribute

Suppose that you want to set the value of the State attribute of the cdc.jmx.SimpleDynamic MBean to the serialized Java object returned by a JMX-MBean Attribute Set job named size.

```
insert_job: change
job_type: JMXTMAS
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver"
mbean_name: "DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic"
mbean_attr: State
jmxt_parameter: payload_job=size
condition: success(size)
```

## mbean\_name Attribute—Specify the Name of the MBean

The mbean\_name attribute specifies the name of an MBean in a JMX job. An MBean is a managed bean (Java object) that represents an application, a device, or any resource that you want to manage.

### Supported Job Types

This attribute is required for the following job types:

- [JMX-MBean Attribute Get \(JMXTMAG\)](#) (see page 232)
- [JMX-MBean Attribute Set \(JMXTMAS\)](#) (see page 233)
- [JMX-MBean Create Instance \(JMXTMC\)](#) (see page 235)
- [JMX-MBean Operation \(JMXTMOP\)](#) (see page 237)
- [JMX-MBean Remove Instance \(JMXTMREM\)](#) (see page 239)
- [JMX-MBean Subscribe \(JMXTSUB\)](#) (see page 240)

### Syntax

This attribute has the following format:

```
mbean_name: "domain_name:key=value[,key=value...]"
```

**"domain\_name:key=value[,key=value...]"**

Specifies the full object name of an MBean.

**domain\_name**

Specifies the default domain name.

**key=value[,key=value...]**

Specifies one or more key=value pairs.

**Limits:** Up to 256 characters; case-sensitive

**Example:** "DefaultDomain:type=SimpleDynamic,index=3"

#### **Example: Query a JMX Server for the Value of an MBean Attribute**

Suppose that you want to know the value of the cachesize attribute for the Config MBean. The URL for the JMX server is service:jmx:rmi:///jndi/rmi://localhost:9999/server, where localhost is the host name and 9999 is the port number.

```
insert_job: query
job_type: JMXML
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_attr: cachesize
```

## mbean\_operation Attribute—Specify the Operation to be Invoked

The mbean\_operation attribute specifies the MBean operation to be invoked in a JMX-MBean Operation job.

### Supported Job Type

This attribute is required for the [JMX-MBean Operation \(JM XMOP\) job type](#) (see page 237).

### Syntax

This attribute has the following format:

`mbean_operation: operation`

#### *operation*

Specifies the operation to be invoked.

**Limits:** Up to 256 characters; case-sensitive

### Example: Invoke an Operation on an MBean

Suppose that you want to invoke the resetmem operation on the config MBean to reset the value of the memory parameter to 50.

```
insert_job: reset
job_type: JM XMOP
machine: agent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_operation: resetmem
jmx_parameter: Integer=50
```

## message\_class Attribute—Specify the Java Class of the JMS Message

The message\_class attribute specifies the Java class of the JMS message in a JMS Publish job.

### Supported Job Type

This attribute is required for the [JMS Publish \(JMSPUB\) job type](#) (see page 228).

### Syntax

This attribute has the following format:

`message_class: class`

***class***

Specifies the Java class of the JMS message.

**Limits:** Up to 256 characters; case-sensitive

**Note:** If the package is not specified, java.lang is assumed.

### Example: Specify Java Class of the JMS Message

This example publishes the message "this is my message" to the queue named Queue. The Java class of the message is String. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere MQ server and 2809 is the ORB port. The job uses the cyberuser JNDI user name to gain access to the connection factory named ConnectionFactory.

```
insert_job: publish
job_type: JMSPUB
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
connection_factory: ConnectionFactory
destination_name: Queue
use_topic: FALSE
message_class: String
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="this is my message"
```

**Note:** The agent does not support JMS messaging on IBM WebSphere. If you have IBM WebSphere MQ, your agent administrator can set up the agent plug-in to run JMS Publish and JMS Subscribe for JMS queues. JMS topics are not supported on IBM WebSphere MQ.

## method\_name Attribute—Specify the Method or Operation to be Invoked

The method\_name attribute specifies the method to be invoked in a job.

### Supported Job Types

This attribute is required for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214) when the operation type is UPDATE
- [POJO](#) (see page 252)
- [RMI \(JAVARMI\)](#) (see page 254)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

This attribute is optional for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

`method_name: method`

#### *method*

Specifies the method to be invoked. In a Session Bean or Entity Bean job, it specifies the method to be invoked on the application server. In a POJO job, it specifies the Java method to call on the instance of the Java object. In an HTTP job, it specifies the path to the servlet to be invoked. In a RMI job, it specifies the method of the remote Java class to invoke.

**Limits:** Up to 256 characters; case-sensitive

**Example:** GetQuote

**Example: Invoke a Method on a Stateless Session Bean**

Suppose that you want to invoke the reverse method on the CybEJBTestBean stateless session bean. The reverse method has one parameter, with type java.lang.String and value a23. The output from the reverse method is saved in the C:\Makapt15 file. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere application server and 2809 is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

## min\_run\_alarm Attribute—Define the Minimum Run Time for a Job

The min\_run\_alarm attribute specifies the minimum run time (in minutes) that a job should require to finish normally. This “reasonability” test can catch errors, such as when an input file is truncated due to an error. If the job runs in less than the specified time, CA Workload Automation AE generates an alarm to alert the operator to take corrective action.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`min_run_alarm: mins`  
***mins***

Defines the minimum number of minutes the job should ever require to finish normally.

**Default:** No minimum set.

**Limits:** This value can be any integer.

**Example:** Define the Minimum Run Time for a Job

This example specifies that the product should generate an alarm if the job finishes in less than two hours:

`min_run_alarm: 120`

## modify\_parameter Attribute—Specify Modify Parameters

The modify\_parameter attribute specifies the modify parameters in an Entity Bean job. You specify modify parameters to update the property values of an existing entity bean. The modify parameters are used by the setter method specified in the method\_name attribute.

### Supported Job Type

When the operation type is UPDATE, this attribute is optional for the [Entity Bean \(ENTRYBEAN\) job type](#) (see page 214).

### Syntax

This attribute has the following format:

```
modify_parameter: type=value | payload_job=job_name[, type=value |
payload_job=job_name...]
```

#### **type=value**

Specifies the Java class and String value for the parameter.

**Limits:** Up to 1024 characters; case-sensitive

**Examples:** String="5:00PM", String[2]="winnt1,winnt2", double=2000

**Note:** If the package of the Java class is not specified, java.lang is assumed.

#### **payload\_job=job\_name**

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

#### **Notes:**

- You can define up to 64 modify parameters using one modify\_parameter attribute or multiple entries of finder\_parameter.
- In each modify\_parameter entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. Specify the parameters in the same order as they appear in the setter method. The job fails if the parameters are listed in the wrong order.

**Example: Update the Phone Number for the Acme Company in a Database**

Suppose that you want to update the phone number for the Acme company to 800-555-0199. The customer entity bean stores the customer ID and phone number. The primary key for the customer is the customer ID. To find the entity bean, the job uses the Acme's customer ID. When the job runs, the Acme company's phone number is changed.

```
insert_job: update
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: UPDATE
method_name: changephone
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
modify_parameter: String="800-555-0199"
```

## monitor\_cond Attribute—Specify a Condition to Monitor For

The monitor\_cond attribute specifies a condition to monitor the database for. The job completes when the condition is met.

### Supported Job Type

This attribute is optional for the [Database Monitor \(DBMON\) job type](#) (see page 207).

### Syntax

This attribute has the following format:

`monitor_cond: condition`

#### *condition*

Specifies the condition to monitor in the database. This condition is equivalent to an SQL where clause.

**Limits:** Up to 500 characters; case-sensitive

### Example: Monitor for an Increase in the Number of Rows in a Table Using a Condition

This example monitors the Inventory\_List table for an increase in the number of rows. If the number of rows increases and the number of units of ProductA has fallen below 1000, the job completes. The database user ID is set to the user who invokes jil to define the job (the default owner).

```
insert_job: dbmon_job
job_type: DBMON
machine: dbagt
tablename: Inventory_List
monitor_type: INCREASE
monitor_cond: ProductA < 1000
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

## monitor\_mode Attribute—Specify Whether to Monitor Conditions Immediately or Continuously

The monitor\_mode attribute specifies whether the job waits until the monitor conditions are met or tries to verify them immediately.

### Notes:

- If you do not specify the monitor\_mode attribute in your OMIP, OMP, or OMS job definition, the job checks for the conditions immediately and completes (the default).
- If you do not specify the monitor\_mode attribute in your OMCPUs, OMD, OMTF, or OMEL job definition, the job waits until the specified conditions are met before completing (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Disk Monitoring \(OMD\)](#) (see page 213)
- [IP Monitoring \(OMIP\)](#) (see page 226)
- [Process Monitoring \(OMP\)](#) (see page 253)
- [Text File Reading and Monitoring \(OMTF\)](#) (see page 274)
- [Windows Event Log Monitoring \(OMEL\)](#) (see page 279)
- [Windows Service Monitoring \(OMS\)](#) (see page 281)

### Syntax

This attribute has the following formats:

- To specify a mode for OMIP, OMP, and OMS jobs:  
`monitor_mode: WAIT | NOW`
- To specify a mode for OMCPUs, OMD, OMTF, and OMEL jobs:  
`monitor_mode: WAIT | NOW | CONTINUOUS`

## **WAIT**

Waits for the conditions to occur. When the conditions are met, the job completes. This is the default for OMCPU, OMD, OMTF, and OMEL jobs.

### **Notes:**

- When using the WAIT value in CPU Monitoring or Disk Monitoring jobs, you must also specify the lower\_boundary attribute, the upper\_boundary attribute, or both.
- When using the WAIT value in Text File Reading and Monitoring jobs, do not specify the upper\_boundary attribute.

## **NOW**

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default for OMIP, OMP, and OMS jobs.

## **CONTINUOUS**

Monitors for the conditions continuously. Each time the specified conditions occur, an alert is written to the scheduler log file (event\_demon.\$AUTOSERV on UNIX and event\_demon.%AUTOSERV% on Windows).

### **Notes:**

- When using the CONTINUOUS value in CPU Monitoring or Disk Monitoring jobs, you must also specify the lower\_boundary attribute, the upper\_boundary attribute, or both.
- When using the CONTINUOUS value in Text File Reading and Monitoring jobs, consider the following points:
  - If the job finds one or more occurrences of the searched text, the first trigger takes place at the first occurrence only. Subsequently, the job only searches for new lines that are added to the text file. In other words, the job resumes the search from the first line following the last line of the file from the previous search.
  - You cannot specify the upper\_boundary attribute.
  - When using the text\_file\_filter\_exists attribute, it must be set to TRUE (the default). The job continues to monitor the file for the existence of the specified text string until the job is completed manually. Each time the text string is found, an alert is triggered.
- This value does not apply to IP Monitoring (OMIP), Process Monitoring (OMP), and Windows Service Monitoring (OMS) jobs.
- To end continuous monitoring, you must complete the job manually by issuing the following command:

```
sendevent -E KILLJOB -J job_name
```

### Example: Monitor a Text File Continuously

This example searches the transmitter.log file for the text string "Warning".

```
insert_job: textfile_job
job_type: OMTF
machine: monagt
text_file_name: /export/home/log/transmitter.log
text_file_filter: Warning
text_file_mode: LINE
lower_boundary: 25
text_file_filter_exists: TRUE
monitor_mode: CONTINUOUS
```

When the job first runs, it searches the content between line 25 and the end of the file. An alert is written to the scheduler log file the first time that the string is found. In other words, suppose that the file contains multiple occurrences of "Warning" between lines 25 to the end of the file, as follows:

```
.
.
.
25
26 Warning
27
28 Warning
29
30
31 Warning
.
.
.
EOF
```

When the job first runs, the trigger only occurs at the first occurrence of the text (line 26). Subsequently, the job continues monitoring only the new data that is *appended* to the file. An alert is triggered each time the string is found in the appended data.

**Note:** Alerts are not triggered for new occurrences of the "Warning" string in the data that has already been searched. For example, suppose that the job has already searched lines 25 to 100 of the file. The file is then modified to include "Warning" on line 30. During continuous monitoring, an alert is *not* triggered for that occurrence.

This job runs until it is completed manually.

#### **Example: Check a Process Status Immediately**

The job in this example checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails.

```
insert_job: omp_unix
job_type: OMP
machine: unixagt
process_name: 2568
process_status: RUNNING
monitor_mode: NOW
```

#### **Example: Monitor an IP Address Until It Stops**

This example monitors a device with DNS name myhost. When the device stops running, the job completes.

```
insert_job: omip_job
job_type: OMIP
machine: monagt
ip_host: myhost
ip_status: STOPPED
monitor_mode: WAIT
```

## monitor\_type Attribute—Specify the Type of Database Change to Monitor For

The monitor\_type attribute specifies the type of database change to monitor for.

**Note:** If you do not specify the monitor\_type attribute in your job definition, the job monitors for an increase or a decrease in the number of rows in the database table (the default).

### Supported Job Type

This attribute is optional for the [Database Monitor \(DBMON\) job type](#) (see page 207).

### Syntax

This attribute has the following format:

mon\_type: INCREASE | DECREASE | VARIANCE

#### INCREASE

Monitors for an increase in the number of rows in the database table.

#### DECREASE

Monitors for a decrease in the number of rows in the database table.

#### VARIANCE

Monitors for an increase or a decrease in the number of rows in the database table.  
This is the default.

**Note:** By default, the specified table is polled every 10 seconds for changes to the number of rows.

### Example: Monitor for an Increase in the Number of Rows in a Table

This example monitors the Inventory\_List table for an increase in the number of rows. By default, the database is checked for changes every 10 seconds. When the number of rows increases, the job completes. This job runs on the dbagt agent computer and the database user ID is set to the user who invokes jil to define the job (the default owner).

```
insert_job: dbmon_job1
job_type: DBMON
machine: dbagt
tablename: Inventory_List
monitor_type: INCREASE
connect_string: "jdbc:db2://172.31.255.255:50000/SAMPLE"
```

## must\_complete\_times Attribute—Specify the Time a Job Must Complete By

The must\_complete\_times attribute defines the time or a list of times that a job must complete by. If a job run does not complete by the specified time, an alarm (MUST\_COMPLETE\_ALARM) is issued.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following formats:

must\_complete\_times: "hh:mm[, hh:mm...]"

must\_complete\_times: +minutes

"**hh:mm[, hh:mm...]**"

Defines one or more absolute times that a job must complete by before an alarm is issued. Specify the time in 24-hour format. Separate each time with a comma.

**Limits:** 00:00-71:59 (2 calendar days ahead of the current calendar day)

#### Notes:

- To define absolute times, you must also specify the following attributes in your job definition:
  - date\_conditions
  - start\_times
- If you specify the start\_mins attribute in the job definition, you can only define relative times. You will get an error if you define absolute times with start\_mins.
- If a job has multiple start times, you must specify the same number of must complete times in the must\_complete\_times attribute. For example, if the job runs at three different times during the day, you must specify three must complete times, corresponding to each run of the job. If the number of must complete times does not match the number of start times, the job terminates.
- The must complete time for a run must be earlier than the start time for the next run. For example, the following values are invalid because the must complete time for the job's second run (11:10) occurs later than the start time of the job's third run (11:00):

start\_times: "10:00, 10:30, 11:00"

must\_complete\_times: "10:10, 11:10, 11:20"

- Absolute times can cross up to 2 calendar days. For example, suppose that you define a job that starts at 11:00 a.m. and you want to specify a must complete time of 10:00 a.m. the next day. This must complete time crosses the day boundary. Because the value of the must complete time is less than the value of the start time, you cannot specify 10:00 a.m. as the must complete time. If 10:00 a.m. is specified, the job issues an error message and terminates the client request. Instead, you must specify the must complete time as 34:00. This time is calculated as follows:

```
must complete time + 24 hours
= 10:00 + 24 hours
= 34:00
```

#### **+minutes**

Defines a relative time in minutes that a job must complete by before an alarm is issued.

**Limits:** +0 to +4319 (2 calendar days from the current calendar day)

#### **Notes:**

- To define a relative time, you must also specify the following attributes in your job definition:
  - date\_conditions
  - start\_mins or start\_timesThe must complete times are calculated relative to the start\_mins or start\_times attributes.
- A relative time can cross up to 2 calendar days. For example, suppose that you define a job that starts at 11:57 p.m. and you want the job to complete within 30 minutes of the start time. The must start time is calculated as follows:

```
start time + relative must complete time
= 11:57 + 30
= 12:27 a.m.
```

This must complete time crosses the day boundary.

#### **Notes:**

- The entire value can be up to 255 characters.
- You cannot use both absolute times and relative times in the same must\_complete\_times attribute. You must use one format for all specified times.
- You *can* specify both must\_start\_times and must\_complete\_times attributes in the same job definition.

### **Example: Specify Absolute Must Complete Times**

This example defines a job to run every day at 10:00 a.m., 11:00 a.m., and 12:00 p.m. The job must complete by 10:08 a.m., 11:08 a.m., and 12:08 p.m. respectively. Otherwise, an alarm is issued for each missed complete time.

```
insert_job: test_must_start_complete
command: /opt/StartTransactions.sh
machine: localhost
date_conditions: y
days_of_week: all
start_times: "10:00, 11:00, 12:00"
must_start_times: "10:02, 11:02, 12:02"
must_complete_times: "10:08, 11:08, 12:08"
```

**Note:** The number of must complete times must match the number of start times. Otherwise, the job terminates. For example, the job terminates if it has one start time and two must complete times, as follows:

```
start_times: "10:00"
must_complete_times: "10:08, 12:08"
```

### **Example: Specify an Absolute Must Complete Time on the Next Day**

This example defines a job to run every day at 12:00 a.m. Suppose that you want each job run to end by 10:12 a.m. the next day. You must specify 34:12 in the must\_complete\_times attribute.

```
insert_job: job3
command: echo "hello"
machine: localhost
date_conditions: y
days_of_week: all
start_times: "12:00"
must_start_times: "34:10"
must_complete_times: "34:12"
```

The must complete time is calculated as follows:

$$\begin{aligned} \text{must complete time} &+ 24 \text{ hours} \\ &= 10:12 + 24 \\ &= 34:12 \end{aligned}$$

### **Example: Specify an Absolute Must Complete Time That Crosses Two Days**

This example defines a job to run every day at 16:00 (4:00 p.m.). Suppose that you want each job run to end by 6:12 a.m. two days later. You must specify 54:12 in the must\_complete\_times attribute.

```
insert_job: job3
job_type: CMD
machine: localhost
command: echo "hello"
date_conditions: y
days_of_week: all
start_times: "16:00"
must_start_times: "54:10"
must_complete_times: "54:12"
```

The must complete time is calculated as follows:

$$\begin{aligned} \text{must complete time} &+ 48 \text{ hours} \\ &= 6:12 + 48 \text{ hours} \\ &= 54:12 \end{aligned}$$

### **Example: Specify Relative Must Complete Times**

This example defines a job to run very day at 10:00 a.m., 11:00 a.m., and 12:00 p.m. Each job run must complete within 8 minutes after each start time (10:08 a.m., 11:08 a.m., and 12:08 p.m.). Otherwise, an alarm is issued for each missed complete time.

```
insert_job: test_must_start_complete
job_type: CMD
machine: localhost
command: /opt/StartTransactions.sh
date_conditions: y
days_of_week: all
start_times: "10:00, 11:00, 12:00"
must_start_times: +3
must_complete_times: +8
```

**Example: Specify Relative Must Complete Times With start\_mins**

This example defines a job to run every day at 10 minute intervals in each hour (for example, 2:00 p.m., 2:10 p.m., 2:20 p.m., and so on). Each job run must complete within 7 minutes after the specified start times. Otherwise, an alarm is issued for each missed complete time. For instance, the 2:10 p.m. job run must complete by 2:17 p.m.

```
insert_job: test_must_start_complete
job_type: CMD
machine: localhost
command: /opt/StartTransactions.sh
date_conditions: y
days_of_week: all
start_mins: 10, 20, 30, 40, 50, 00
must_start_times: +2
must_complete_times: +7
```

**Example: Specify an Unmatched Number of Must Start and Must Complete Times**

Suppose that you want to define a job to run twice. You specify two start times and only one must start time and one must complete time. The job is *not* inserted into the database because the number of must start times and must complete times must match the number of start times.

```
insert_job: job2
job_type: CMD
machine: localhost
command: echo "Hello"
date_conditions: y
days_of_week: all
start_times: "12:00, 13:00"
must_start_times: "12:01"
must_complete_times: "12:02"
```

To successfully insert the job definition, you must specify two must start times and two must complete times, as follows:

```
insert_job: job2
job_type: CMD
machine: localhost
command: echo "Hello"
date_conditions: y
days_of_week: all
start_times: "12:00, 13:00"
must_start_times: "12:01, 13:01"
must_complete_times: "12:02, 13:02"
```

CHK\_START and CHK\_COMPLETE events for the must start times and must complete times are inserted to database. The job's first run occurs at 12:00 p.m. If the job runs for three minutes, the MUST\_START\_ALARM is *not* generated because the job started by the specified must start time (12:01). However, the MUST\_COMPLETE\_ALARM is issued because the job ran past the specified must complete time (12:02).

## must\_start\_times Attribute—Specify the Time a Job Must Start By

The must\_start\_times attribute defines the time or a list of times that a job must start by. If the job does not start by the specified time, an alarm (MUST\_START\_ALARM) is issued.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following formats:

must\_start\_times: "hh:mm[, hh:mm...]"

must\_start\_times: +minutes

"hh:mm[, hh:mm...]"

Defines one or more absolute times that a job must start by before an alarm is issued. Specify the time in 24-hour format. Separate each time with a comma.

**Limits:** 00:00-71:59 (2 calendar days ahead of the current calendar day)

#### Notes:

- To define absolute times, you must also specify the following attributes in your job definition:
  - date\_conditions
  - start\_times
- If you specify the start\_mins attribute in the job definition, you can only define relative times. You will get an error if you define absolute times with start\_mins.
- If a job has multiple start times, you must specify the same number of must start times in the must\_start\_times attribute. For example, if the job runs at three different times during the day, you must specify three must start times, corresponding to each run of the job. If the number of must start times does not match the number of start times, the job terminates.
- The must start time for a run must be earlier than the start times for the next run. For example, the following values are invalid because the must start time for the job's second run (11:10) occurs later than the start time of the job's third run (11:00):

start\_times: "10:00, 10:30, 11:00"

must\_start\_times: "10:10, 11:10, 11:20"

- Absolute times can cross up to 2 calendar days. For example, suppose that you define a job that starts at 11:00 a.m. and you want to specify a must start time of 10:00 a.m. the next day. This must start time crosses the day boundary. Because the value of the must start time is less than the value of the start time, you cannot specify 10:00 a.m. as the must start time. If 10:00 a.m. is specified, the job issues an error message and terminates the client request. Instead, you must specify the must start time as 34:00. This time is calculated as follows:

```
must start time + 24 hours
= 10:00 + 24 hours
= 34:00
```

#### **+minutes**

Defines a relative time in minutes that a job must start by before an alarm is issued.

**Limits:** +0 to +4319 (2 calendar days from the current calendar day)

#### **Notes:**

- To define a relative time, you must also specify the following attributes in your job definition:
  - date\_conditions
  - start\_mins or start\_timesThe must start times are calculated relative to the start\_mins or start\_times attributes.
- A relative time can cross up to 2 calendar days. For example, suppose that you define a job that starts at 11:57 p.m. and you want the job to start within 30 minutes of the start time. The must start time is calculated as follows:

```
start time + relative must start time
= 11:57 + 30
= 12:27 a.m.
```

This must start time crosses the day boundary.

#### **Notes:**

- The entire value can be up to 255 characters.
- You cannot use both absolute times and relative times in the same must\_start\_times attribute. You must use one format for all specified times.
- You *can* specify both must\_start\_times and must\_complete\_times attributes in the same job definition.

### Example: Specify Absolute Must Start Times

This example defines a job to run every day at 10:00 a.m., 11:00 a.m., and 12:00 p.m. The job must start by 10:02 a.m., 11:02 a.m., and 12:02 p.m. respectively. Otherwise, an alarm is issued for each missed start time.

```
insert_job: test_must_start_complete
command: /opt/StartTransactions.sh
machine: localhost
date_conditions: y
days_of_week: all
start_times: "10:00, 11:00, 12:00"
must_start_times: "10:02, 11:02, 12:02"
must_complete_times: "10:08, 11:08, 12:08"
```

**Note:** The number of must start times must match the number of start times. Otherwise, the job terminates. For example, the job terminates if it has one start time and two must start times, as follows:

```
start_times: "10:00"
must_start_times: "10:02, 12:02"
```

### Example: Specify an Absolute Must Start Time on the Next Day

This example defines a job to run every day at 12:00 a.m. Suppose that you want each job run to start by 10:10 a.m. the next day. You must specify 34:10 in the must\_start\_times attribute.

```
insert_job: job3
command: echo "hello"
machine: localhost
date_conditions: y
days_of_week: all
start_times: "12:00"
must_start_times: "34:10"
must_complete_times: "34:12"
```

The must start time is calculated as follows:

```
must start time + 24 hours
= 10:10 + 24 hours
= 34:10
```

### **Example: Specify an Absolute Must Start Time That Crosses Two Days**

This example defines a job to run every day at 16:00 (4:00 p.m.). Suppose that you want each job run to start by 6:10 a.m. two days later. You must specify 54:10 in the `must_start_times` attribute.

```
insert_job: job3
job_type: CMD
machine: localhost
command: echo "hello"
date_conditions: y
days_of_week: all
start_times: "16:00"
must_start_times: "54:10"
must_complete_times: "54:12"
```

The must start time is calculated as follows:

$$\begin{aligned} \text{must start time} &+ 48 \text{ hours} \\ &= 6:10 + 48 \text{ hours} \\ &= 54:10 \end{aligned}$$

### **Example: Specify Relative Must Start Times**

This example defines a job to run every day at 10:00 a.m., 11:00 a.m., and 12:00 p.m. Each job run must start within 3 minutes after each start time (10:03 a.m., 11:03 a.m., and 12:03 p.m.). Otherwise, an alarm is issued for each missed start time.

```
insert_job: test_must_start_complete
job_type: CMD
machine: localhost
command: /opt/StartTransactions.sh
date_conditions: y
days_of_week: all
start_times: "10:00, 11:00, 12:00"
must_start_times: +3
must_complete_times: +8
```

#### **Example: Specify Relative Must Start Times With start\_mins**

This example defines a job to run every day at 10 minute intervals in each hour (for example, 2:00 p.m., 2:10 p.m., 2:20 p.m., and so on). Each job run must start within 2 minutes after the specified start times. Otherwise, an alarm is issued for each missed start time. For instance, the 2:10 p.m. job must start by 2:12 p.m.

```
insert_job: test_must_start_complete
job_type: CMD
machine: localhost
command: /opt/StartTransactions.sh
date_conditions: y
days_of_week: all
start_mins: 10, 20, 30, 40, 50, 00
must_start_times: +2
must_complete_times: +7
```

#### **Example: Specify a Relative Must Start Time on the Next Day**

This example specifies a relative must start time of 12:01 a.m. the next day. The must start time is calculated by adding three minutes to the start time. The calculated time crosses into the next day.

```
insert_job: job2
job_type: CMD
machine: localhost
command: echo "Hello"
date_conditions: y
days_of_week: all
start_times: "23:58"
must_start_times: +3
```

#### **Example: Specify an Unmatched Number of Must Start and Must Complete Times**

Suppose that you want to define a job to run twice. You specify two start times and only one must start time and one must complete time. The job is *not* inserted into the database because the number of must start times and must complete times must match the number of start times.

```
insert_job: job2
job_type: CMD
machine: localhost
command: echo "Hello"
date_conditions: y
days_of_week: all
start_times: "12:00, 13:00"
must_start_times: "12:01"
must_complete_times: "12:02"
```

To successfully insert the job definition, you must specify two must start times and two must complete times, as follows:

```
insert_job: job2
job_type: CMD
machine: localhost
command: echo "Hello"
date_conditions: y
days_of_week: all
start_times: "12:00, 13:00"
must_start_times: "12:01, 13:01"
must_complete_times: "12:02, 13:02"
```

CHK\_START and CHK\_COMPLETE events for the must start times and must complete times are inserted to database. The job's first run occurs at 12:00 p.m. If the job runs for three minutes, the MUST\_START\_ALARM is *not* generated because the job started by the specified must start time (12:01). However, the MUST\_COMPLETE\_ALARM is issued because the job ran past the specified must complete time (12:02).

## n\_retrys Attribute—Define the Number of Times to Restart a Job After a Failure

The n\_retrys attribute specifies how many times to attempt to restart the job after it exits with a FAILURE status. If a job exits with a TERMINATED status, it does not restart.

This attribute applies to application failures (for example, CA Workload Automation AE cannot find a file or a command, permissions are not properly set, and so on). It does not apply to system or network failures (for example, when a computer is unavailable, socket connection time-outs, failure of a fork in the agent, failure of the file system space resource check, and so on).

### UNIX Notes:

- Job restarts after system or network failures are controlled by the MaxRestartTrys parameter in the configuration file.
- The delay between restarts is determined by the RestartConstant and RestartFactor parameters in the configuration file, up to the maximum specified by the MaxRestartWait parameter.

### Windows Notes:

- Job restarts after system or network failures are controlled by the setting in the Max Restart Trys field on the Scheduler - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator.
- The delay between restarts is determined by the Restart Constant and Restart Factor settings on the Scheduler - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator, up to the maximum specified by the Max Restart Wait setting.

### Supported Job Types

This attribute is optional for all job types.

## Syntax

This attribute has the following format:

`n_retrys: attempts`

### ***attempts***

Defines the number of times to attempt to restart the job after it exits with a FAILURE status.

**Limits:** This value can be any integer in the range 0 to 20.

**Default:** 0

**Note:** The n\_retrys value applies when the command associated with the job fails. When a job fails for other reasons, such as invalid credentials or unavailable space, the MaxRestartTrys value in the configuration file defines how many times the job should be restarted. When the the MaxRestartTrys value is zero, the job is not restarted.

## **Example: Set the Job to Automatically Restart Up to Five Times**

This example sets the job to automatically restart up to five times after an application failure:

`n_retrys: 5`

This means that the job would start as scheduled, and if it fails, it would restart up to five times for a total of six attempts.

## no\_change Attribute—Define the Minimum Change Required in CPU or Disk Usage

The no\_change attribute defines the value that the CPU or disk usage must change by to trigger the job. If the change in CPU or disk usage is within this specified value, the job does not trigger.

### Supported Job Types

This attribute is optional for the following job types:

- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Disk Monitoring \(OMD\)](#) (see page 213)

**Note:** To use this attribute with OMD jobs, you must specify WAIT or CONTINUOUS for the monitor\_mode attribute. This attribute does not apply to OMD jobs when the monitor mode is NOW.

### Syntax

This attribute has the following format:

- To specify a no\_change value for CPU usage:

    no\_change: *delta\_percent*

*delta\_percent*

        Defines the percentage value that the CPU usage must change to trigger the job.

**Limits:** 1-100

- To specify a no\_change value for disk usage:

    no\_change: *delta\_value*

*delta\_value*

        Defines the value that the CPU usage must change by to trigger the job. The unit for this value is specified by the disk\_format attribute.

**Limits:** Up to 10 characters

**Example:** 100

**Note:** The job will trigger only if the following conditions are met:

- The change value is within the range specified by the lower\_boundary and upper\_boundary attributes.
- The change value is greater than the delta specified by the last scanned amount. That is, the first time CPU or disk usage matches the job criteria, an alert is written to the scheduler log file. Subsequently, an alert is triggered only if the change in CPU or disk usage is greater than the delta calculated using the last scanned amount that was registered in a trigger.

#### **Example: Monitor for a Change in CPU Usage Greater Than 10 Percent**

This example continuously monitors the CPU available on the winagt computer. An alert is written to the scheduler log file each time the available CPU is less than 25 percent or greater than 75 percent, and the CPU usage changes by more than 10 percent. If the change in value is less than or equal to 10, the job does not register a change.

```
insert_job: omcpu_job
job_type: OMCPU
machine: winagt
cpu_usage: FREE
lower_boundary: 25
upper_boundary: 75
inside_range: FALSE
no_change: 10
monitor_mode: CONTINUOUS
```

#### **Example: Monitor for a Change in Disk Usage Greater than 100 KB**

This example continuously monitors the available disk space in kilobytes (KB) on the local Windows C drive. When the available space is in the 35000000 to 36000000 KB range, the first alert is written to the scheduler log file.

Subsequently, an alert is triggered each time the available disk space is within the specified boundaries and the disk usage changes by more than 100 KB. If the amount of change is less than or equal to 100 KB, the job does not register a change.

```
insert_job: omd_job
job_type: OMD
machine: winagent
disk_drive: C
disk_space: FREE
disk_format: KB
lower_boundary: 35000000
upper_boundary: 36000000
inside_range: TRUE
no_change: 100
monitor_mode: CONTINUOUS
```

The following table shows four sequential scans:

| Scan | Scanned Amount (Kilobytes) | Does the Trigger Occur?                                                                                                                                          |
|------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | 35018896                   | Yes.                                                                                                                                                             |
| 2    | 35018900                   | No. Comparing scan 2 to scan 1, the delta value is only 4 KB. This scanned amount will not be included in the next calculation.                                  |
| 3    | 35018795                   | Yes. Comparing scan 3 to scan 1, the delta value is greater than 100 KB. The delta value of the next scan will be calculated using the scan 3 value of 35018795. |
| 4    | 36000001                   | No. The scanned amount is outside the lower and upper boundary range.                                                                                            |

## notification\_id Attribute—Identify the Recipient of the Notification

The notification\_id attribute identifies the recipient of the notification generated when you have set the send\_notification attribute to **y**, **1**, or **f** and the job you are defining completes with the appropriate status.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`notification_id: recipient`

#### *recipient*

Specifies the notification ID or email address of the user who should receive the notification.

**Limits:** Up to 255 alphanumeric characters

### Example: Identify the Recipient of the Notification

This example defines a user to whom the message is to be sent:

`notification_id: ccsuser1`

## notification\_msg Attribute—Define the Message to Include in the Notification

The notification\_msg attribute defines the message to include in the notification generated when you have set the send\_notification attribute to **y**, **1**, or **f** and the job you are defining completes with the appropriate status.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`notification_msg: value`

#### **value**

Defines a message to include in the notification.

**Limits:** Up to 255 alphanumeric characters

### Example: Define a Message to Include in a Notification

This example defines a message to include in a notification:

`notification_msg: The server appears to be down.`

## one\_way Attribute—Specify Whether to Invoke the Service One Way

The one\_way attribute specifies whether the job completes without waiting for a response after the agent invokes the operation.

**Note:** If you do not specify the one\_way attribute in your job definition, the job waits for a response after the agent invokes the operation before completing (the default).

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

`one_way: TRUE | FALSE`

**TRUE**

Invokes the service one way. After the agent invokes the operation, the Web Service job completes without waiting for a response.

**FALSE**

Does not invoke the service one way. This is the default.

**Example: Invoke a Web Service Operation**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webserviceX.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a `java.lang.String` object, which maps to the XML type string in the return namespace <http://www.webserviceX.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## operation\_type Attribute—Specify the Operation Type

The operation\_type attribute indicates the operation to perform on the entity bean in an Entity Bean job. The Entity Bean job lets you create an entity bean, update the property values of an existing entity bean, or remove an entity bean from the database.

### Supported Job Type

This attribute is required for the [Entity Bean \(ENTYBEAN\) job type](#) (see page 214).

### Syntax

This attribute has the following format:

operation\_type: CREATE | UPDATE | REMOVE

#### CREATE

Creates an entity bean that is stored in a relational database on your application server.

#### UPDATE

Updates the property values of an entity bean instance in a relational database on your application server.

#### REMOVE

Removes an instance of an entity bean stored in a relational database on your application server.

### Notes:

- To update an entity bean, you require the finder\_name, finder\_parameter, method\_name, and modify\_parameter attributes.
- To remove an entity bean, you require the finder\_name and finder\_method attributes.

#### Example: Remove an Entity Bean

Suppose that you want to remove the customer record for the Acme customer. The record is stored in the database by the customer ID. When the job runs, the row in the customer table that corresponds to the Acme customer ID is removed.

```
insert_job: remove
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
operation_type: REMOVE
finder_name: findByPrimaryKey
finder_parameter: String="customerid"
```

## oracle\_appl\_name Attribute—Specify the Name of an Oracle Applications Application

The oracle\_appl\_name attribute specifies the name of the Oracle Applications application that the single request or request set belongs to.

This attribute must be used with the oracle\_appl\_name\_type attribute.

#### Supported Job Type

This attribute is required for the following job types:

- [Oracle Applications Request Set \(OASET\)](#) (see page 244)
- [Oracle Applications Single Request \(OASG\)](#) (see page 246)

#### Syntax

This attribute has the following format:

oracle\_appl\_name: *short\_name* | *display\_name*

#### *short\_name*

Specifies the short name of the Oracle Applications application. The short name is defined by the Oracle Applications Concurrent Manager.

**Limits:** Up to 50 characters; case-sensitive

**Note:** The value of the oracle\_appl\_name\_type attribute must be SHORT.

***display\_name***

Specifies the display name of the Oracle Applications application. In Oracle Applications, the display name is part of the request definition and is found in the Application field.

**Limits:** Up to 256 characters; case-sensitive

**Note:** The value of the oracle\_appl\_name\_type attribute must be DISPLAY.

**[Example: Specify the Display Name of an Oracle Applications Application](#)**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The history of the program is included in the description.

```
insert_job: oasg_disp
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_desc: CONFIRMED REQ 2010808 by TYB212
```

**[Example: Specify the Short Name of an Oracle Applications Application](#)**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

## oracle\_appl\_name\_type Attribute—Specify the Type of Oracle Applications Name

The oracle\_appl\_name\_type attribute specifies whether the name of the Oracle E-Business Suite application is the short name or the display name.

### Notes:

- This attribute must be used with the oracle\_appl\_name attribute.
- The default type is SHORT. If you do not specify the oracle\_appl\_name\_type attribute in your job definition, the job interprets the oracle\_appl\_name value to be a short name.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_appl\_name\_type: SHORT | DISPLAY

#### SHORT

Indicates that the Oracle E-Business Suite application is specified by its short name. The short name is defined by the Oracle Applications Concurrent Manager. This is the default.

#### DISPLAY

Indicates that the Oracle E-Business Suite application is specified by its display name.

#### **Example: Specify the Display Name of an Oracle E-Business Suite Application**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The history of the program is included in the description.

```
insert_job: oasg_disp
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_desc: CONFIRMED REQ 2010808 by TYB212
```

#### **Example: Specify the Short Name of an Oracle E-Business Suite Application**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

## **oracle\_args Attribute—Define Argument Values to Pass to a Single Request**

The oracle\_args attribute defines the argument values to pass to an Oracle E-Business Suite single request. The argument values override the default values that are defined by the registered Oracle Applications Concurrent Manager program.

### **Supported Job Type**

This attribute is optional for the [Oracle E-Business Suite Single Request \(OASG\) job type](#) (see page 246).

## Syntax

This attribute has the following format:

`oracle_args: argument[,argument...]`

### ***argument***

Defines each argument value to pass to an Oracle Applications single request. In Oracle Applications, arguments are part of the program definition and are found in the Concurrent Program Parameters dialog.

### **Notes:**

- You can specify up to 100 arguments. Separate each argument with a comma.
- The entire value can be up to 500 characters.
- Do not add a space after a comma unless the argument value starts with a space. If a value starts with a space, enclose the value in single quotation marks.
- To use a default value defined in Oracle E-Business Suite, enter one comma in the argument position and specify Y in the oracle\_use\_arg\_def attribute.

**Example:** X22F,,X56R143,'X23T'

### **Example: Specify Argument Values**

Suppose that you want to pass the argument values, T,*DefArg2*,X23,,*DefArg5*, to a single request program. The job uses the argument values that are specified in the job definition and the default values defined in Oracle E-Business Suite as follows:

- The first argument, T, and the third argument, X23, are specified in the job definition.
- The second argument, *DefArg2*, and the fifth argument, *DefArg5*, are defined as defaults in Oracle E-Business Suite.
- The fourth argument is not specified in the job definition or defined as a default on Oracle E-Business Suite, so the agent passes an empty string for that argument.

```
insert_job: oasg_args
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_program: FNDSCARU
oracle_use_arg_def: Y
oracle_args: T,,X23,,
```

## oracle\_desc Attribute—Describe a Single Request Job

The oracle\_desc attribute defines a description for an Oracle E-Business Suite Single Request job.

### Supported Job Type

This attribute is optional for the [Oracle E-Business Suite Single Request \(OASG\) job type](#) (see page 246).

### Syntax

This attribute has the following format:

`oracle_desc: description`

#### *description*

Defines a description for an Oracle E-Business Suite Single Request job. The description displays in the Name column of the Requests dialog.

**Limits:** Up to 30 characters; case-sensitive

#### Notes:

- Your agent administrator can specify a default description for all Oracle E-Business Suite Single Request jobs by setting the oa.default.desc parameter in the agent's agentparm.txt file.
- The oracle\_desc attribute overrides the default description specified in the agent's agentparm.txt.

### Example: Specify a Description for an Oracle E-Business Suite Single Request Job

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The history of the program is included in the description.

```
insert_job: oasg_disp
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_desc: CONFIRMED REQ 2010808 by TYB212
```

## oracle\_mon\_children Attribute—Specify Whether to Monitor Children Programs

The oracle\_mon\_children attribute specifies whether the children of the Oracle E-Business Suite programs are monitored. Program children are programs that are released by the parent program.

**Note:** To use this attribute, you must also specify the oracle\_mon\_children\_delay attribute in your job definition.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Copy Single Request \(OACOPY\)](#) (see page 242)
- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_mon\_children: Y | N

**Y**

Specifies that the children of the Oracle E-Business Suite programs are monitored.

**N**

Specifies that the children of the Oracle E-Business Suite programs are not monitored. This is the default.

**Note:** Up to five levels of children can be monitored for each job.

**Example: Specify that the Children of a Single Request Program are Monitored**

This example runs a request set named FNDRSSUB873. The request set belongs to the application with the short name FND in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The children of the single request program are monitored.

```
insert_job: OA7
job_type: OASET
machine: localhost
oracle_appl_name: FND
oracle_appl_name_type: SHORT
oracle_req_set: FNDRSSUB873
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_mon_children: Y
oracle_mon_children_delay: 10
oracle_save_output: Y
oracle_use_arg_def: Y
```

## **oracle\_mon\_children\_delay Attribute—Define the Time to Wait Before Monitoring Children Programs**

The oracle\_mon\_children\_delay attribute defines the number of seconds to wait after an Oracle E-Business Suite program completes before monitoring its children.

**Note:** To use this attribute, you must also specify the oracle\_mon\_children attribute in your job definition.

### **Supported Job Types**

This attribute is optional for the following job types:

- [Oracle E-Business Suite Copy Single Request \(OACOPY\)](#) (see page 242)
- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### **Syntax**

This attribute has the following format:

```
oracle_mon_children_delay: seconds
```

***seconds***

Defines the number of seconds to wait after an Oracle E-Business Suite program completes before monitoring its children.

**Limits:** 0-99999

**Example:** 60

**Note:** When the oracle\_mon\_children\_delay attribute is defined in an Oracle E-Business Suite Request Set job, the setting is applied to all the programs in the request set. You cannot specify a different setting for each program.

**[Example: Delay Monitoring Children by 60 Seconds](#)**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The children of the single request program are monitored 60 seconds after the program completes.

```
insert_job: oasg_nomon
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_mon_children: Y
oracle_mon_children_delay: 60
```

## oracle\_print\_copies Attribute—Specify the Number of Copies to Print

The oracle\_print\_copies attribute specifies the number of copies to print for an Oracle E-Business Suite single request or request set.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

`oracle_print_copies: num_copies`  
***num\_copies***

Specifies the number of copies to print. In Oracle E-Business Suite, the number of copies is specified as a request definition option and is found in the Copies column of the Upon Completion dialog.

**Limits:** Up to 6 numeric digits

**Example:** 100

### Notes:

- In an Oracle E-Business Suite Request Set job, you can specify the number of copies to print for an individual program in the request set by using the oracle\_programdata attribute. This attribute overrides the oracle\_print\_copies attribute for that particular program.
- Your agent administrator can specify a default number of copies for all Oracle Applications jobs by setting the oa.default.printCopies parameter in the agent's agentparm.txt file. The oracle\_print\_copies attribute overrides the default number of copies specified in the agent's agentparm.txt.

**Example: Print 15 Copies of an Oracle E-Business Suite Single Request**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The job specifies 15 copies to be printed on the Q\_DEVELOPMENT printer in PORTRAIT style.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_printer:Q_DEVELOPMENT
oracle_print_style: PORTRAIT
oracle_print_copies: 15
```

## oracle\_print\_style Attribute—Specify a Print Style

The oracle\_print\_style attribute specifies an Oracle E-Business Suite print style.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_print\_style: *print\_style*

#### *print\_style*

Specifies a print style. The value must match the name of an Oracle Applications print style. In Oracle Applications, the print style is specified as a request definition option and is found in the Style field of the Upon Completion dialog.

**Limits:** Up to 30 characters; case-sensitive

**Example:** PORTRAIT

**Notes:**

- Your agent administrator can specify a default print style for all Oracle E-Business Suite Request Set and Single Request jobs by setting the oa.default.printStyle parameter in the agent's agentparm.txt file.
- The oracle\_print\_style attribute overrides the default print style specified in the agent's agentparm.txt.
- In an Oracle E-Business Suite Request Set job, you can specify the print style for an individual program in the request set by using the oracle\_programdata attribute. This attribute overrides the oracle\_print\_style attribute and the oa.default.printStyle default parameter for that particular program.

**Example: Specify an Oracle E-Business Suite Print Style**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The job specifies 15 copies to be printed on the Q\_DEVELOPMENT printer in PORTRAIT style.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_printer:Q_DEVELOPMENT
oracle_print_style: PORTRAIT
oracle_print_copies: 15
```

## oracle\_printer Attribute—Specify a Printer

The oracle\_printer attribute specifies the name of a printer to be used by Oracle E-Business Suite.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

`oracle_printer: printer_name`

#### *printer\_name*

Specifies the name of a printer that Oracle Applications can print to. This printer must be defined in Oracle Applications. In Oracle Applications, the printer name is specified as a request definition option and is found in the Printer column of the Upon Completion dialog.

**Limits:** Up to 30 characters; case-sensitive

**Note:** Enclose paths that contain delimiters (such as spaces) in quotation marks.

**Example:** "`\\\printer path\K6700`"

### Notes:

- Your agent administrator can specify a default printer for all Oracle E-Business Suite Request Set and Single Request jobs by setting the oa.default.printer parameter in the agent's agentparm.txt file.
- The oracle\_printer attribute overrides the default printer specified in the agent's agentparm.txt.
- In an Oracle E-Business Suite Request Set job, you can specify the printer for an individual program in the request set by using the oracle\_programdata attribute. This attribute overrides the oracle\_printer attribute and the oa.default.printer default parameter for that particular program.

#### **Example: Specify a Printer for Oracle E-Business Suite**

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The job specifies 15 copies to be printed on the Q\_DEVELOPMENT printer in PORTRAIT style.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_printer:Q_DEVELOPMENT
oracle_print_style: PORTRAIT
oracle_print_copies: 15
```

## oracle\_program Attribute—Specify the Name of a Single Request Program

The oracle\_program attribute specifies the short name of an Oracle E-Business Suite single request program.

### Supported Job Type

This attribute is required for the [Oracle E-Business Suite Single Request \(OASG\) job type](#). (see page 246)

### Syntax

This attribute has the following format:

oracle\_program: *program\_short\_name*

#### *program\_short\_name*

Specifies the short name of the Oracle Applications single request program. The program short name must be registered in the Oracle Applications Concurrent Manager. In Oracle Applications, the program short name is part of the program definition and is found in the Short Name field of the Concurrent Programs dialog.

**Limits:** Up to 30 characters; case-sensitive

### Example: Specify the Name of an Oracle E-Business Suite Single Request Program

This example runs a single request program named FNDSCARU. The single request belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility. The history of the program is included in the description.

```
insert_job: oasg_disp
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_desc: CONFIRMED REQ 2010808 by TYB212
```

## oracle\_programdata Attribute—Specify Data for a Program in a Request Set

The oracle\_programdata attribute specifies data for an individual program in an Oracle E-Business Suite request set. This data overrides the print parameters specified for the entire request set.

### Supported Job Type

This attribute is optional for the [Oracle E-Business Suite Request Set \(OASET\) job type](#) (see page 244).

### Syntax

This attribute has the following format:

```
oracle_programdata: index=number, args="argument[,argument...]",
printer=printer_name, print_style=print_style, print_copies=num_copies, saveop=Y |
N
```

#### **index=program\_number**

Specifies the order number of a program in an Oracle Applications request set.

**Limits:** Integer

**Example:** 1 (specifies the first program in the request set)

#### **args="argument[,argument...]"**

Defines the argument values to override the default values that are defined by the registered Oracle Applications Concurrent Manager program.

**Limits:** Up to 100 argument values, for a total of 500 characters

#### **Notes:**

- You can specify multiple arguments. Separate each argument with a comma.
- Do not add a space after a comma unless the argument value starts with a space. If a value starts with a space, enclose the value in single quotation marks.
- To use a default value defined in Oracle Applications, enter one comma in the argument position and specify Y in the oracle\_use\_arg\_def attribute.
- The entire value can be up to 4096 characters.

**Example:** args="X22F,,X56R143,,X23T"

**printer=***printer\_name*

Specifies the name of a printer that Oracle Applications can print to. This printer must be defined in Oracle Applications. In Oracle Applications, the printer name is specified as a request definition option and is found in the Printer column of the Upon Completion dialog.

**Limits:** Up to 30 characters; case-sensitive

**Note:** Enclose paths that contain delimiters (such as spaces) in quotation marks.

**Example:** "\\\printer path\K6700"

**print\_style=***print\_style*

Specifies a print style. The value must match the name of an Oracle Applications print style. In Oracle Applications, the print style is specified as a request definition option and is found in the Style field of the Upon Completion dialog.

**Limits:** Up to 30 characters; case-sensitive

**Example:** PORTRAIT

**print\_copies=***num\_copies*

Specifies the number of copies to print. In Oracle E-Business Suite, the number of copies is specified as a request definition option and is found in the Copies column of the Upon Completion dialog.

**Limits:** Up to 6 numeric digits

**Example:** 100

**Default:** 0

**saveop=Y | N**

**Y**

Saves output from an Oracle Applications program. In Oracle Applications, this option is specified as a request definition option and corresponds to selecting the Save all Output Files check box of the Upon Completion dialog.

**N**

Does not save output from an Oracle Applications program. In Oracle Applications, this option is specified as a request definition option and corresponds to clearing the Save all Output Files check box of the Upon Completion dialog.

**Note:** To specify data for multiple programs in a request set, define a separate oracle\_programdata attribute for each program.

### Example: Specify Data for Two Programs in an Oracle E-Business Suite Request Set

Suppose that you want to run an Oracle E-Business Suite request set named EXTRACTS on the oaagent agent. The request set belongs to the application with the display name Application Object Library in Oracle E-Business Suite. The job definition specifies the following default settings for all programs in the request set:

- The number of copies to be printed is 1.
- The print style is LANDSCAPE.
- The printer is \\printer path\Q8.

The first oracle\_programdata attribute overrides the default values for the first program in the request set. The first program uses the arguments T23 and R1 and prints two copies using the \\printer path\Q1 printer in PORTRAIT style.

The second oracle\_programdata attribute overrides the default values for the fifth program in the request set. The fifth program uses arguments R and R1 and prints three copies using the \\printer path\Q2 printer in PORTRAIT style.

The other programs in the request set use the default values.

```
insert_job: oaset_prog
job_type: oaset
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_req_set: EXTRACTS
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_printer: "\\printer path\Q8"
oracle_print_style: LANDSCAPE
oracle_print_copies: 1
oracle_programdata: index=1, args="T23,,R1", printer="\\printer\Q1",
print_style=PORTRAIT, print_copies=2, saveop=Y
oracle_programdata: index=5, args="R,R1,", printer="\\printer\Q2",
print_style=PORTRAIT, print_copies=3, saveop=N
```

## oracle\_req\_set Attribute—Specify the Name of a Request Set

The oracle\_req\_set attribute specifies the short name of an Oracle E-Business Suite request set.

### Supported Job Type

This attribute is required for the [Oracle E-Business Suite Request Set \(OASET\) job type](#) (see page 244).

### Syntax

This attribute has the following format:

oracle\_req\_set: *request\_set\_short\_name*

***request\_set\_short\_name***

Specifies the short name of the Oracle Applications request set. In Oracle Applications, the request set short name is part of the Request Set definition and is found in the Set Code field of the Request Set dialog.

**Limits:** Up to 30 characters; case-sensitive

### Example: Specify the Name of an Oracle E-Business Suite Request Set

This example runs a request set named FNDRSSUB1310 on the oaagent agent. The request set belongs to the application with the short name BIS in Oracle E-Business Suite. The job runs under the SYSADMIN user with Business Intelligence Super User, Progress S&L responsibility. The job's output is saved.

```
insert_job: oaset_resp
job_type: oaset
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: BIS
oracle_req_set: FNDRSSUB1310
oracle_user: SYSADMIN
oracle_resp: "Business Intelligence Super User, Progress S&L"
oracle_save_output: Y
```

## oracle\_resp Attribute—Specify a Responsibility Name

The oracle\_resp attribute specifies an Oracle E-Business Suite responsibility name.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Copy Single Request \(OACOPY\)](#) (see page 242)
- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_resp: *responsibility\_name*

***responsibility\_name***

Specifies an Oracle Applications responsibility name.

**Limits:** Up to 100 characters; case-sensitive

**Single Request or Request Set Job Notes:**

- If you do not specify the oracle\_resp attribute in the job definition, a default responsibility name must be defined in the oa.default.responsibility parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- The oracle\_resp attribute overrides the default responsibility name specified in the agent's agentparm.txt file.

**Copy Single Request Job Notes:**

- To override the responsibility name specified in the existing single request, both the user name and responsibility name are required:
  - If a default user name is not specified in the oa.default.user parameter in the agent's agentparm.txt file, you must specify the oracle\_user attribute in the job definition.
  - If a default responsibility name is not specified in the oa.default.responsibility parameter in the agent's agentparm.txt file, you must specify the oracle\_resp attribute in the job definition.
- If the user name or responsibility name is not specified in either the job definition or in the agentparm.txt file, the job copies the user name and responsibility name from the original single request.

**Example: Specify an Oracle E-Business Suite Responsibility Name**

This example runs a request set named FNDRSSUB1310 on the oaagent agent. The request set belongs to the application with the short name BIS in Oracle E-Business Suite. The job runs under the SYSADMIN user with Business Intelligence Super User, Progress S&L responsibility. The job's output is saved.

```
insert_job: oaset_resp
job_type: oaset
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: BIS
oracle_req_set: FNDRSSUB1310
oracle_user: SYSADMIN
oracle_resp: "Business Intelligence Super User, Progress S&L"
oracle_save_output: Y
```

### Example: Copy a Single Request Job and Override the Responsibility Name

Suppose that you want to copy an existing single request defined on Oracle E-Business Suite. In this example, the job copies the single request job with request ID 2255470 and overrides the Oracle E-Business Suite user name and responsibility name defined in the agentparm.txt file.

```
insert_job: oacopy_single
job_type: oacopy
machine: oaagent
request_id: 2255470
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

## oracle\_save\_output Attribute—Specify Whether to Save Output

The oracle\_save\_output attribute specifies whether to save the output from an Oracle E-Business Suite Single Request or Request Set job.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_save\_output: Y | N

**Y**

Saves output from an Oracle Applications program. In Oracle Applications, this option is specified as a request definition option and corresponds to selecting the Save all Output Files check box of the Upon Completion dialog.

**N**

Does not save output from an Oracle Applications program. In Oracle Applications, this option is specified as a request definition option and corresponds to clearing the Save all Output Files check box of the Upon Completion dialog. This is the default.

**Note:** In an Oracle E-Business Suite Request Set job, you can specify whether to save the output for an individual program in the request set by using the oracle\_programdata attribute. This attribute overrides the oracle\_save\_output attribute for that particular program.

### Example: Save the Output of an Oracle E-Business Suite Request Set Job

This example runs a request set named FNDRSSUB1310 on the oaagent agent. The request set belongs to the application with the short name BIS in Oracle E-Business Suite. The job runs under the SYSADMIN user with Business Intelligence Super User, Progress S&L responsibility. The job's output is saved.

```
insert_job: oaset_resp
job_type: oaset
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: BIS
oracle_req_set: FNDRSSUB1310
oracle_user: SYSADMIN
oracle_resp: "Business Intelligence Super User, Progress S&L"
oracle_save_output: Y
```

## oracle\_use\_arg\_def Attribute—Specify Whether to Use Argument Defaults

The oracle\_use\_arg\_def attribute specifies whether to use default values for arguments that not defined using the oracle\_args attribute or the oracle\_programdata attribute. The default arguments are defined in Oracle E-Business Suite.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

oracle\_use\_arg\_def: Y | N

**Y**

Specifies that the job uses the default values defined in Oracle Applications for arguments that are not specified in the job definition.

**N**

Specifies that the job does not use the default values defined in Oracle Applications for arguments that are not specified in the job definition. This is the default.

**Example: Use Oracle E-Business Suite Argument Defaults**

Suppose that you want to pass the argument values, T,*DefArg2*,X23,,*DefArg5*, to a single request program. The job uses the argument values that are specified in the job definition and the default values defined in Oracle E-Business Suite as follows:

- The first argument, T, and the third argument, X23, are specified in the job definition.
- The second argument, *DefArg2*, and the fifth argument, *DefArg5*, are defined as defaults in Oracle E-Business Suite.
- The fourth argument is not specified in the job definition or defined as a default on Oracle E-Business Suite, so the agent passes an empty string for that argument.

```
insert_job: oasg_args
job_type: oasg
machine: oaagent
oracle_appl_name_type: DISPLAY
oracle_appl_name: Application Object Library
oracle_user: SYSADMIN
oracle_resp: System Administrator
oracle_program: FNDSCARU
oracle_use_arg_def: Y
oracle_args: T,,X23,,
```

## oracle\_user Attribute—Specify a User Name

The oracle\_user attribute specifies an Oracle E-Business Suite user name that the job runs under.

### Supported Job Types

This attribute is optional for the following job types:

- [Oracle E-Business Suite Copy Single Request \(OACOPY\)](#) (see page 242)
- [Oracle E-Business Suite Request Set \(OASET\)](#) (see page 244)
- [Oracle E-Business Suite Single Request \(OASG\)](#) (see page 246)

### Syntax

This attribute has the following format:

`oracle_user: user_name`

***user\_name***

Specifies the Oracle Applications user name that the job runs under.

**Limits:** Up to 100 characters; case-sensitive; it cannot contain delimiters (such as spaces)

#### Single Request or Request Set Job Notes:

- If you do not specify the oracle\_user attribute in the job definition, a default user name must be defined in the oa.default.user parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- The oracle\_user attribute overrides the default user name specified in the agent's agentparm.txt.

#### Copy Single Request Job Notes:

- To override the user name specified in the existing single request, both the user name and responsibility name are required:
  - If a default user name is not specified in the oa.default.user parameter in the agent's agentparm.txt file, you must specify the oracle\_user attribute in the job definition.
  - If a default responsibility name is not specified in the oa.default.responsibility parameter in the agent's agentparm.txt file, you must specify the oracle\_resp attribute in the job definition.
- If the user name or responsibility name is *not* specified in either the job definition or in the agentparm.txt file, the job copies the user name and responsibility name from the original single request.

### Example: Specify an Oracle E-Business Suite User Name

This example runs a single request program named FNDSCARU. The single request belongs to the application with the short name ACCOUNTS in Oracle E-Business Suite. The job runs under the SYSADMIN user with System Administrator responsibility.

```
insert_job: oasg_short
job_type: oasg
machine: oaagent
oracle_appl_name_type: SHORT
oracle_appl_name: ACCOUNTS
oracle_program: FNDSCARU
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

### Example: Copy a Single Request Job and Override the User Name

Suppose that you want to copy an existing single request defined on Oracle E-Business Suite. In this example, the job copies the single request job with request ID 2255470 and overrides the Oracle E-Business Suite user name and responsibility name defined in the agentparm.txt file.

```
insert_job: oacopy_single
job_type: oacopy
machine: oaagent
request_id: 2255470
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

## owner Attribute—Define the Owner of the Job

The owner attribute defines the owner of the job. By default, the owner attribute is set to *authenticated\_user@host*, where *authenticated\_user* is the operating system user who invokes jil to define the job. If your job requires a different user ID, you must override the owner value. Otherwise, the job does not run successfully.

For jobs that run on other software, such as PeopleSoft and databases, the owner can be the user ID associated with and authenticated by the software. For example, SAP jobs run under SAP user IDs. As another example, an FTP job requires an FTP user ID to connect to the FTP server.

You must define user IDs and their corresponding passwords to CA Workload Automation AE by using the autosys\_secure command. In a job definition, you can use the owner attribute to specify the user ID. CA Workload Automation AE retrieves the corresponding password from the database.

### Notes:

- If CA Workload Automation AE is running in native security mode, you can change the owner value only if you have EDIT superuser permissions.
- If CA Workload Automation AE is running in external security mode using CA EEM, you can change the owner value only if you have as-owner authority.
- CA Workload Automation AE uses the owner value for all job types except for File Trigger. CA Workload Automation AE does *not* use the oscomponent.default.user parameter located in the agent's agentparm.txt file.

### Supported Job Types

This attribute is optional for all job types except for File Trigger (FT) jobs.

**Note:** File Trigger jobs run under the user that runs the agent (when the jobs run as threads), or they run under the oscomponent.default.user value (when the jobs run as external processes).

## UNIX Syntax

This attribute has the following format:

*owner: user@machine | user*

***user@machine | user***

Specifies a valid user with an account on the specified real machine. The user must have an account on all machines where the job can run.

**Default:** *user@machine*, where *user* is the user who invokes jil to define the job and *machine* is the real machine that user was logged on to.

### Notes:

- The specified owner owns and has edit permission for all jobs defined during the session. The UNIX command specified in the job runs under the owner's user ID. When a command starts on the agent, the uid of the process is changed to the owner of the job.
- Only the specific user on the specific machine can edit the job. For the job to run, this *user@machine* combination must have execute permission for the UNIX command specified in the job, on the client for the job.
- If you used the autosys\_secure command to activate remote authentication, the user's permission on the agent is verified at job run time. To do this, the agent makes a ruserok() system call, which verifies the agent's /etc/hosts.equiv and the user's .rhosts files to validate that the requesting user is registered in that environment. This is a "local" verification; it is not related to rshd or rlogind, which rely on the configuration of the agent's /etc/services and /etc/inetd.conf files. If the rshd and rlogind are disabled on a client, but the /etc/hosts.equiv and the .rhosts files are configured correctly, users cannot rlogin or rsh to the client computer, but they can run jobs on it.

## Windows Syntax

This attribute has the following format:

`owner: user@host_or_domain | user`

`user@host_or_domain | user`

Identifies a valid user with an account on the specified local domain host or Windows network domain, which must be a real machine. The user must have an account on all machines where the job can run, and the user must have a valid password for that account in the database.

**Default:** `user@host_or_domain`, where `user` is the user who invokes jil to define the job and `host_or_domain` is the host or domain that user was logged on to.

### Notes:

- To run a job, the agent attempts to log on to the client as the owner (`user@host_or_domain`). If logon is unsuccessful, the agent attempts to log on to the client as the user specified by the owner attribute at the host specified in the machine attribute (`user@host`). For either of these logon attempts to work, the Windows user IDs and passwords must exist in the database. If agent user authentication is enabled, it is performed on the second (`user@host`) logon attempt. In this case, the proper Trusted Hosts or Trusted Users permissions are also required.
- For jobs to run on the Primary Domain Controller in a Windows domain, the job owner must be `user@domain` (`user@host` will not work). On Windows, you cannot log on to the domain controller machine. Instead, you must log on to the domain when using the domain controller machine.
- For Command jobs, the command runs as the user currently logged on. After the user logs on to a machine, that user must also have all the necessary Windows permissions to access the command and any resources needed to run it.
- If you change your computer name on the Identification tab of the Windows Control Panel Network dialog, you must change the Host Name on the DNS tab of the TCP/IP Protocol dialog to the same name. CA Workload Automation AE uses the Host Name as the host portion of the owner and ignores the Computer Name.

**Notes:**

- When defining the job, the owner can grant other users edit or execute permission for that job. Execute permission controls which users can issue sendevent commands, such as STARTJOB or KILLJOB on the job. However, this does not affect whose permissions the job's command runs under.
- To determine the job owner, issue the following command at an instance command prompt:  
`autorep -J job_name -q`
- The EDIT superuser can use the update\_job subcommand to change the owner of an individual job. To change a large number of jobs, the EDIT superuser can use the autorep command to dump multiple job definitions to an output file, change the owner, and reload the changed job definitions using the jil command.

**Example: Define the Owner of a Job**

This example changes the owner of the job to chris. Only the EDIT superuser can change the value. The user chris on any machine in the network can edit the job. The job's command runs with the permissions of chris.

```
owner: chris
```

## Changing the Owner in Multiple Jobs

To change a large number of jobs, the EDIT superuser can invoke the autorep command to dump multiple job definitions to an output file, change the owner, and reload the changed job definitions using the jil command. The following example shows how to save all job definitions to a file:

```
autorep -J ALL -q > dump_file
```

The output of this command is formatted exactly as a job definition script as shown in the following example:

```
insert_job: test_job
job_type: c
command: sleep 60
machine: juno
owner: jerry@jupiter
permission: wx
alarm_if_fail: 1
```

The owner field of each job definition is typically commented out, unless the EDIT superuser runs the autorep command to generate the report. This is because only the EDIT superuser can change the owner field. After generating this report, the EDIT superuser can use a text editor to change the owner field and reload the job definitions into the database using the jil command, as follows:

```
jil < dump_file
```

## Job Owners and Security

CA Workload Automation Agent for UNIX, Linux, or Windows provides a local security feature that controls which users are allowed to submit jobs on behalf of other users. However, this security rule does not apply to CA Workload Automation AE. On CA Workload Automation AE, jobs are always submitted to run under the user specified in the owner attribute. If local security is enabled on the agent, the agent checks the permissions of the job owner only. The agent does not check the CA Workload Automation AE user who submits the job.

**Note:** For more information about how CA Workload Automation AE works with local security on the agent, see the *UNIX Implementation Guide* or *Windows Implementation Guide*.

## permission Attribute—Specify the Users with Edit and Execute Permissions

The permission attribute specifies which users have edit and execute permissions for the job you are defining.

**UNIX Note:** The permission attribute is based on the same permissions used in native UNIX. It uses the user ID (uid), and group ID (gid) from the UNIX environment to control who can edit job definitions and who can execute the actual command specified in the job.

**Windows Note:** The permission attribute provides users (by type) with edit and execute permissions for a specific job. By default, only a job's owner has edit and execute permissions on a job.

**Note:** When issuing commands or defining jobs that run on a different operating system (for example, Windows to UNIX or UNIX to Windows), you must use the syntax appropriate to the operating system where the job will run.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`permission: permission [, permission]`

#### *permission*

Specifies the comma-delimited permission levels to associate with the job. The order in which *permission* values are specified is not important.

**Limits:** This value can be up to 30 alphanumeric characters in length.

Valid values are:

**gx**

(UNIX only) Assigns group execute permissions to the job.

**ge**

(UNIX only) Assigns group edit permissions to the job.

**me**

Indicates that any authorized user may edit the job, regardless of the machine they are on. Otherwise, the user must be logged on to the machine specified in the owner field (for example, *user@host\_or\_domain*).

**mx**

Indicates that any authorized user may execute the job, regardless of the machine they are on. Otherwise, the user must be logged on to the machine specified in the owner field (for example, *user@host\_or\_domain*).

**we**

Assigns world edit permissions to the job.

**wx**

Assigns world execute permissions to the job.

The job's owner always has full edit and execute permissions.

**Default:** Machine permissions are turned off.

**Example: Set Permissions to Execute and Edit a Job on UNIX**

This example sets permissions on UNIX so that any user can execute the job, but only members of your group can edit it:

```
permission: ge, wx
```

**Example: Set Permissions to Execute a Job on Windows**

This example sets permissions on Windows so that any authorized user can execute the job regardless of the machine they are logged on to:

```
permission: mx
```

## User Types

CA Workload Automation AE supports the following types of users for any job:

**Owner**

Indicates the user who created the job.

**World**

Indicates any user.

**Group**

(UNIX only) Indicates any user who is in the same group as the owner.

**Note:** Windows does not support the UNIX group permission attribute.

The default owner is the user who initiated JIL to define the job. The owner always has both edit and execute permissions for the job on the machine on which it was defined. Permission attributes can extend edit and execute capability to other users and other machines.

## Permission Types on UNIX

On UNIX, CA Workload Automation AE supports multiple levels of permissions associated with a job. Every job has the following levels of permissions:

### Edit

Indicates that the user can edit, override, or delete the job definition.

### Execute

Indicates that the user can affect how the job runs, typically by issuing a sendevent command. Events that affect how a job runs are:

- CHANGE\_STATUS
- DELETEJOB
- FORCE\_STARTJOB
- JOB\_OFF\_HOLD
- JOB\_OFF\_ICE
- JOB\_ON\_HOLD
- JOB\_ON\_ICE
- KILLJOB
- SEND\_SIGNAL
- STARTJOB

The job owner has edit permission on the job, and the UNIX command specified in the job runs under the owner's user ID.

When a command runs on the machine specified in the job definition, the setuid(uid) system call changes the uid of the process to that of the job's owner.

## Permission Types on Windows

On Windows, CA Workload Automation AE supports the following permission levels:

### Edit

Indicates that the user can edit, override, or delete the job definition.

### Exec

Indicates that the user can affect how the job runs, typically by issuing a sendevent command. Events that affect how a job runs are:

- CHANGE\_STATUS
- FORCE\_STARTJOB
- JOB\_OFF\_HOLD
- JOB\_OFF\_ICE
- JOB\_ON\_HOLD
- JOB\_ON\_ICE
- KILLJOB
- STARTJOB

### Machine

Indicates that, by default, all edit and execute permissions are valid only on the machine on which the job was defined. Permission attributes can extend this permission to other machines.

## User and Permission Types on UNIX

### Valid on UNIX

When a job is created, CA Workload Automation AE retrieves the user ID from the environment and associates it with the job. Then, the product uses the owner's current umask value to supply default permissions to the job. The product uses the umask "write" permission as the default "edit" permission for the job, and it uses the umask "execute" permission as the default "execute" permission for the job.

## Job Permissions and Windows

If you are defining jobs and running them on different operating systems, keep the following in mind:

- When you define a job to run on a Windows client, you can set group permissions, but they are ignored. Group permissions are only used when a job is edited or executed on a UNIX computer.
- When you edit a job from a Windows computer, the group edit permission is ignored. In this case, the user editing the job must be the job's owner or World Edit permissions must be specified for the job.
- When you run a job from a Windows computer, the group execute permission is ignored. In this case, the user executing the job must be the job's owner or World Execute permissions must be specified for the job.

## poll\_interval Attribute—Define the Frequency to Poll CPU or Disk Usage

The poll\_interval attribute defines how often the CPU or disk usage is polled.

**Note:** If you are monitoring continuously and you do not specify the poll\_interval attribute in your job definition, the CPU or disk usage is polled every 10 seconds (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Disk Monitoring \(OMD\)](#) (see page 213)

**Note:** To use this attribute, you must specify WAIT or CONTINUOUS for the monitor\_mode attribute. This attribute does not apply when the monitor mode is NOW.

## Syntax

This attribute has the following format:

**poll\_interval: *seconds***

***seconds***

Defines the interval (in seconds) between successive scans of the CPU or disk usage.

**Default:** 10

**Limits:** Up to 10 digits

### Notes:

- Your agent administrator can specify a default poll interval for all CPU and Disk Monitoring jobs by setting the objmon.scaninterval parameter in the agent's agentparm.txt file. That agent parameter is specified in milliseconds.
- The poll\_interval attribute overrides the default poll interval specified in the agent's agentparm.txt.

### **Example: Poll CPU Usage Every 60 Seconds**

This example continuously monitors the CPU available on the unixagent computer. The job polls the CPU usage every 60 seconds. Each time the available CPU is within 75 and 95 percent, an alert is written to the scheduler log file. The job continues monitoring the CPU usage until it is ended manually.

```
insert_job: omcpu_job
job_type: OMCPU
machine: unixagent
cpu_usage: FREE
inside_range: TRUE
poll_interval: 60
monitor_mode: CONTINUOUS
lower_boundary: 75
upper_boundary: 95
```

## port\_name Attribute—Specify the Port Name Within the Namespace

The port\_name attribute specifies the WSDL port name within the target namespace in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

port\_name: *portname*

#### *portname*

Specifies the WSDL port name within the target namespace. A WSDL port describes the operations exposed by a web service and defines the connection point to the web service.

**Limits:** Up to 100 characters; case-sensitive

**Note:** In a Web Service job, if you specify the WSDL\_URL attribute but not the endpoint\_URL attribute, you must specify both the service\_name and port\_name attributes. For the job to run successfully without the endpoint\_URL attribute, the agent must be running on the same computer as the application server such as WebLogic or JBoss. If you specify both the WSDL\_URL and endpoint\_URL attributes, then the service\_name and port\_name attributes are optional.

#### **Example: Specify the WSDL Port Name for Getting Stock Quotes**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webservicex.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a `java.lang.String` object, which maps to the XML type string in the return namespace <http://www.webservicex.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## priority Attribute—Define the Queue Priority of the Job

The priority attribute defines the queue priority of the job you are defining.

When a job is ready to run on a designated machine but the current load on that machine is too large to accept the new job's load, CA Workload Automation AE queues the job for that machine so it runs when sufficient resources are available.

The queue priority establishes the relative priority of all jobs queued for a machine. A lower number indicates a higher priority. Higher priority jobs completely block lower priority jobs from running until all the high priority jobs have run. Therefore, if the available machine load units allow a lower priority job to start, but there are higher priority jobs that are queued for machine load units, the lower priority job is queued. The lower priority job will run after machine load units are available and all higher priority jobs have run.

Although you cannot queue box jobs, you can assign them priorities. This permits jobs in boxes to run according to their priority, instead of in the order in which they were defined, which is the default.

**Note:** You cannot use the priority or job\_load attribute if you specify a user-defined load balancing script in the machine attribute.

### Supported Job Types

This attribute is optional for all job types. This attribute does not apply to box jobs.

### Syntax

This attribute has the following format:

**priority: *priority\_level***

#### ***priority\_level***

Defines the queue priority of the job. The lower the value, the higher the priority; therefore, 1 is the highest possible queued priority. 0 signifies to run the job immediately, regardless of the current machine load

**Limits:** Any integer 0 or greater

**Default:** 0

#### **Example: Set the Job to Run Immediately**

This example sets the job to always run immediately, regardless of the current load on the client:

```
priority: 0
```

#### **Example: Set the Job to Run with Highest Priority**

This example sets the job to run with the highest priority without overriding the machine load control mechanism:

```
priority: 1
```

#### **Example: Set the Job to Run in the Background**

This example sets the job to run in the background when the machine load is low:

```
priority: 100
```

## **process\_name Attribute—Specify the Name of the Process to be Monitored**

The process\_name attribute specifies the name of the process to be monitored.

### **Supported Job Type**

This attribute is required for the [Process Monitoring \(OMP\) job type](#) (see page 253).

### **UNIX and Windows Syntax**

This attribute has the following format:

```
process_name: process_name
```

#### ***process\_name***

Specifies the name of the process to be monitored.

**UNIX:** Specify a PID or process name.

**Windows:** Specify a full path or process name.

**Limits:** Up to 256 characters; case-sensitive

## i5/OS Syntax

This attribute has the following format:

`process_name: jobnumber/username/jobname`

### *jobnumber*

Specifies the six-digit job number or \*ALL.

### *username*

Specifies the user ID the job runs under. The value can be a generic name or \*ALL.

**Limits:** Up to 10 characters

### *jobname*

Specifies the job name on the i5/OS system. The value can be a generic name or \*ALL.

**Limits:** Up to 10 characters

### Notes:

- If you specify only the PID or process name and the job runs on an i5/OS computer, the agent searches the active UNIX workload for a matching PID or process name.
- On i5/OS, you can use generic names to specify the process name. A generic name starts with characters that are part of a valid name and ends with an asterisk (\*). The asterisk denotes any number of characters and can only be placed at the end of a generic name. A name cannot contain a single asterisk and no other characters. To specify all possible names, use the special value \*ALL.
- On all operating systems, if you have multiple processes on a system with the same process name, the job completes successfully as follows:

| Status and Monitor Mode                       | Job Completion Requirements                                                                               |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| process_status: RUNNING<br>monitor_mode: NOW  | The job completes successfully if at least one of the processes with the specified name is running.       |
| process_status: RUNNING<br>monitor_mode: WAIT | The job completes successfully when at least one of the processes with the specified name starts running. |
| process_status: STOPPED<br>monitor_mode: NOW  | The job completes successfully if all the processes with the specified name are stopped.                  |
| process_status: STOPPED<br>monitor_mode: WAIT | The job completes successfully when all the processes with the specified name stops running.              |

### **Example: Monitor Multiple Instances**

This example monitors the Microsoft SQL Server processes of two instances using the full path name. When the server process stops, the job monitoring that instance completes successfully. The first job monitors the sqlserver.exe process in the ...\\MSSQL.1\\MSSQL\\Binn directory. The second job monitors the sqlserver.exe process in the ...\\MSSQL.2\\MSSQL\\Binn directory.

```
insert_job: mon_sql_server_instance1
job_type: OMP
machine: mssqlserver
process_name: "C:\\Program Files\\Microsoft SQL
Server\\MSSQL.1\\MSSQL\\Binn\\sqlservr.exe"
process_status: stopped
monitor_mode: wait

insert_job: mon_sql_server_instance2
job_type: OMP
machine: mssqlserver
process_name: "C:\\Program Files\\Microsoft SQL
Server\\MSSQL.2\\MSSQL\\Binn\\sqlservr.exe"
process_status: stopped
monitor_mode: wait
```

### **Example: Monitor the Agent Process on i5/OS**

This example monitors the CYBAGENT process on an i5/OS computer. The job checks the process status immediately and completes successfully if the process is running.

```
insert_job: omp_i5_onejob
job_type: OMP
machine: i5agt
process_name: 123456/CYBESPU/CYBAGENT
process_status: RUNNING
monitor_mode: NOW
```

### **Example: Monitor i5/OS Processes That Have Similar Names**

This example monitors all processes running on an i5/OS computer under the JDOE user profile and whose names start with CALC. When all of these processes stop running, the job completes successfully.

```
insert_job: omp_i5
job_type: OMP
machine: i5agt
process_name: *ALL/JDOE/CALC*
process_status: STOPPED
monitor_mode: WAIT
```

#### Example: Monitor an i5/OS Process That Has Any Job Number and Is Running Under Any User Name

This example monitors for any process named CYBAGENT, regardless of the first part (job number) and the second part (user name) of the process ID. The job completes successfully if at least one process named CYBAGENT is running. The entire process ID is enclosed in quotation marks so that the text following /\* is not interpreted as a comment.

```
insert_job: omp_i5_all
job_type: OMP
machine: i5agt
process_name: "*ALL/*ALL/CYBAGENT"
process_status: RUNNING
monitor_mode: NOW
```

#### Example: Monitor a UNIX Process on an i5/OS Computer

This example monitors for the BMPROC process in the PASE environment on an i5/OS computer. If the BMPROC process is running, the job completes.

```
insert_job: omp_i5_unix
job_type: OMP
machine: i5agt
process_name: BMPROC
process_status: RUNNING
monitor_mode: NOW
```

## process\_status Attribute—Specify the Status of the Process to be Monitored

The process\_status attribute specifies the status of the process to be monitored. The job checks whether the status of the process matches the status specified by this attribute.

**Note:** If you do not specify the process\_status attribute in your job definition, the job checks whether the process is stopped (the default).

### Supported Job Type

This attribute is optional for the [Process Monitoring \(OMP\) job type](#) (see page 253).

## Syntax

This attribute has the following format:

process\_status: RUNNING | STOPPED

### RUNNING

Specifies that the job monitors the process for a running status.

### STOPPED

Specifies that the job monitors the process for a stopped status. This is the default.

#### [Example: Monitor a Running Process](#)

This example monitors the process with ID 2568. If the process is running, the job completes successfully. If the process is not running, the job continues monitoring it until the process starts running.

```
insert_job: omp_unix
job_type: OMP
machine: unixagt
process_name: 2568
process_status: RUNNING
monitor_mode: WAIT
```

#### [Example: Monitor a Stopped Process](#)

This example monitors the cybAgent.exe process. The job checks the process status immediately and completes successfully if the process is stopped. If the process is running, the job fails.

```
insert_job: omp_win
job_type: OMP
machine: winagt
process_name: "c:\Program files\Agent\cybAgent.exe"
process_status: STOPPED
monitor_mode: NOW
```

## profile Attribute—Specify a Job Profile

The profile attribute specifies a profile that defines the non-system environment variables that a job uses. The variables in the profile are sourced before the job runs.

System environment variables are the only other variables defined for the command's execution. Therefore, you must define non-system environment variables in a user-defined job profile. System environment variables are set before profile variables. Therefore, you can use references to system environment variables in job profiles.

**Note:** If a command that typically runs when entered at the command line fails when run as a job, it is usually due to the incomplete specification of the required environment for the command in the job's profile.

### Supported Job Types

This attribute is optional for the following job types:

- [Command \(CMD\)](#) (see page 204)
- [File Watcher \(FW\)](#) (see page 219), if the job is submitted to the legacy agent

## UNIX Syntax

This attribute has the following format:

**profile: *path\_name***

***path\_name***

Defines the full path to and name of the job profile. The job profile must be a shell script.

**Limits:** Up to 255 characters

**Note:** You cannot use variable substitution in this value.

### Notes:

- Job profiles on UNIX are always sourced using the job owner's default shell, which is set for the user in the etc/passwd file. Therefore, when you create a job profile, you must use the syntax of the owner's default shell. For example, if the owner's default shell is the Korn shell, you must use Korn syntax in the profile script.
- CA Workload Automation AE searches for the specified profile on the machine where the command is to run. The agent always spawns a process and starts the shell in that process, passing it the name of the profile to source. This profile typically includes the definitions and exports of environment variables, which can be referenced in the job's command. This is helpful if the command is a shell script.
- Redirection of the standard input, standard output, and standard error files will probably fail.
- You must set the \$PATH variable in the profile, because CA Workload Automation AE uses it to locate the command specified in the job.
- For Korn shell users, we recommend that you explicitly set any other required environment variables in the shell script that is specified as the command to run. Alternatively, you can source additional shell scripts from your main shell script.
- If you want to set permissions for std\_out and std\_err to -rw-r--r--, you must set umask 022 set it in the specified profile. If you do not set umask 022, the standard output and standard error files will have world write permissions.

## Windows Syntax

This attribute has the following format:

`profile: profile_name`

***profile\_name***

Specifies the name of a job profile. You can specify a profile that is located on the computer where the job runs. Alternatively, you can specify the path to a profile file on a remote machine, as follows:

`\\machine_name\share_name\profile_name.txt`

**Limits:** Up to 255 characters

### Notes:

- The owner of the job must have read access to the specified profile.
- On Windows, you create job profiles using the Job Profiles window in the CA Workload Automation AE Administrator utility. These profiles contain variable=value pairs that define the environment variables. The profiles are stored in the SystemAgent\agent\_name\profiles directory of the CA Workload Automation AE computer. If you move a job profile to another location, you must specify the full path when you assign the profile to a job. For more information about creating, viewing, and deleting job profiles on Windows, see the *Online Help* for the Administrator utility.

## Example: Assign a Job Profile on UNIX

This example assigns the my\_profile profile to a Command job. The profile is located in the user's home directory, /usr/home.

```
insert_job: test_run
job_type: CMD
machine: unixagent
command: /bin/touch /tmp/test_run.out
profile: /usr/home/my_profile
```

## Example: Assign a Job Profile on Windows

This example assigns the eng profile to a Command job. The eng profile was defined in the Job Profiles window in the CA Workload Automation AE Administrator utility.

```
insert_job: test_run
job_type: CMD
machine: winagent
command: "c:\bin\test.bat"
profile: eng
```

**Example: Assign a Job Profile that is Located on Another Windows Machine**

This example assigns the cmdprof.txt profile to a Command job. The cmdprof.txt profile is defined on the dev machine, which is different from the machine where the job runs.

```
insert_job: test_run
job_type: CMD
machine: winagent
command: "c:\bin\test.bat"
profile: \\dev\share\cmdprof.txt
```

## **provider\_url Attribute—Specify a Service Provider URL (EJB and JMS Jobs)**

The provider\_url attribute specifies the URL of the service provider using dotted decimal notation or DNS name in an Entity Bean, Session Bean, JMS Publish, and JMS Subscribe job. The service provider implements a context or initial context. This context can be plugged in dynamically to the JNDI architecture used by the JNDI client.

### **Supported Job Types**

This attribute is required for the following job types:

- [Entity Bean \(ENTYBEAN\)](#) (see page 214)
- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)
- [Session Bean \(SESSBEAN\)](#) (see page 270)

## Syntax

This attribute has the following format:

`provider_url: "location"`

**"location"**

Specifies the JNDI service provider URL.

- For WebLogic servers, the format is the following:

`"t3://WLIPAddress[:ORBPort]"`

*WLIPAddress*

Specifies the IP address or domain name of the WebLogic Application Server.

*ORBPort*

(Optional) Specifies the ORB port.

**Default:** 7001

### Examples:

- `"t3://localhost:7001"`
- `"t3://172.24.36.107:7001"` (IPv4)
- `"t3://0:0:0:0:FFFF:192.168.00.00:7001"` (IPv6)

- For WebSphere servers, the format is the following:

`"iiop://WSIPAddress[:ORBPort]"`

*WSIPAddress*

Specifies the IP address or domain name of the WebSphere Application Server.

*ORBPort*

(Optional) Specifies the ORB port.

**Default:** 2809

### Examples:

- `"iiop://172.24.0.0:2809"`
- `"iiop://172.24.36.107:2809"` (IPv4)
- `"iiop://0:0:0:0:FFFF:192.168.00.00:2809"` (IPv6)

**Limits:** Up to 256 characters; case-sensitive

### **Example: Specify the Service Provider URL for a WebLogic Application Server**

Suppose that you want to create an entity bean that stores information about a customer such as the customer ID and phone number. The initial context factory supplied by the JNDI service provider is `weblogic.jndi.WLInitialContextFactory`. The service provider's URL is `t3://localhost:7001`, where `localhost` is the domain name of the WebLogic application server and `7001` is the ORB port. When the job runs, the entity bean instance is created.

```
insert_job: create
job_type: ENTYBEAN
machine: appagent
initial_context_factory: weblogic.jndi.WLInitialContextFactory
provider_url: "t3://localhost:7001"
bean_name: customer
create_name: createcustomer
operation_type: CREATE
create_parameter: String="customerid", String="800-555-0100"
```

### **Example: Specify the Service Provider URL for a WebSphere Application Server**

Suppose that you want to invoke the reverse method on the `CybEJBTestBean` stateless session bean. The reverse method has one parameter, with type `java.lang.String` and value `a23`. The output from the reverse method is saved in the `C:\Makapt15` file. The initial context factory supplied by the JNDI service provider is `com.ibm.websphere.naming.WsnInitialContextFactory`. The service provider's URL is `iiop://172.24.0.0:2809`, where `172.24.0.0` is the IP address of the WebSphere application server and `2809` is the ORB port. When the job runs, the output of the reverse method is stored in the output destination file.

```
insert_job: reverse
job_type: SESSBEAN
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
bean_name: CybEJBTestBean
method_name: reverse
destination_file: "C:\Makapt15"
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="a23"
```

## provider\_url Attribute—Specify a Host URL (HTTP Jobs)

The provider\_url attribute specifies the host where the program you want to invoke resides in a HTTP job.

### Supported Job Type

This attribute is required for the [HTTP job type](#) (see page 222).

### Syntax

This attribute has the following format:

```
provider_url: "servlet_url"
"servlet_url"
```

Specifies the host where the program or servlet you want to invoke resides. The URL has the following format:

"http://host[:port][/action]"

#### host

Specifies the name of the computer running the application server.

#### port

(Optional) Specifies the port the host uses to listen for HTTP requests.

**Default:** 80

#### action

(Optional) Specifies the path to the servlet to be invoked.

**Limits:** Up to 256 characters; case-sensitive

**Example:** "http://localhost/cgi-bin/test.sh"

**Note:** HTTP and HTTPS are supported.

### Notes:

- If you omit the servlet path (action) in this attribute, you must specify the method\_name attribute in the job definition. If the method\_name attribute is omitted from the job definition, the agent assumes that the provider\_url attribute specifies the full path to the servlet.
- If you specify the servlet path (action) in this attribute, do not include the method\_name attribute in the job definition. If the method\_name attribute is included in the job definition, the agent assumes that the provider\_url attribute specifies only the server host name.

**Example: Specify the Host URL in an HTTP Job**

Suppose that you want to define a job to perform a Google search and have the results returned to the job's spool file. You also want to refine your search results by specifying a filter. In this example, the job uses the HTTP GET method to perform the Google search on "ca workload automation". When the job runs, the job's spool file includes all matches that contain the filter AE.

```
insert_job: google
job_type: HTTP
machine: appagent
invocation_type: GET
provider_url: "http://google.com/search"
j2ee_authentication_order: BASIC,DIGEST,NTLM
filter: .*AE.*
j2ee_parameter: q="ca workload automation"
```

## ps\_args Attribute—Pass Additional Parameters for the PeopleSoft Report

The ps\_args attribute specifies additional parameters to pass to a PeopleSoft report.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_args: argument ...`

#### *argument*

Specifies additional argument values to pass to the PeopleSoft report.

**Limits:** Case-sensitive

**Note:** You can pass multiple strings as a single argument by enclosing them in quotation marks: “parm1 parm2”

#### Notes:

- The entire value can be up to 254 characters.
- Specify multiple arguments by separating them with blank spaces.
- To pass an argument with delimiters (such as spaces and semicolons), enclose the argument in quotation marks.
- If you specify more than one ps\_args attribute, the second ps\_args attribute within the job definition overrides the first, with the exception of a blank character string, which is ignored.

**Example: Ignore Parameters in the PS\_PRCDEFN Table and Pass Additional Parameters to the nVision Report**

In this example, the job parameters are not updated with data in the PS\_PRCDEFN table. The ps\_args attribute passes additional parameters to the nVision report.

```
insert_job: pass_args
job_type: PS
machine: psagt
owner:VP1@server01
ps_process_name: NVSRUN
ps_process_type: nVision-Report
ps_dest_type: FILE
ps_dest_format: XLS
ps_output_dest: "c:\test\testnv.xls"
ps_skip_parm_updates: Yes
ps_args: "-NRNVARIABLE -NBUAUS01
-NHLhttp://10.1.1.40/psp/ps/EMPLOYEE/ERP/c/REPORT_BOOKS.IC_RUN_DRILLDOWN.GBL?Action=A"
```

## ps\_dest\_format Attribute—Specify the Output Format for a PeopleSoft Report

The ps\_dest\_format attribute specifies the output format for the PeopleSoft report.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_dest\_format: *output\_format*

#### *output\_format*

Specifies the field name of the output destination format. PeopleSoft stores the list of output destination formats in the PSXLATITEM table. This value corresponds to the Format field in PeopleSoft.

**Limits:** Up to 8 characters; case-sensitive; it cannot contain delimiters (such as spaces)

**Note:** Do not use the following special characters: left parenthesis ( ( ), right parenthesis ( ) ), and apostrophe (''). Use caution when using other special characters, such as backslash (\) and at (@).

Format options are the following:

| Field Number | Field Name | Description                   |
|--------------|------------|-------------------------------|
| 1            | ANY        | Any                           |
| 2            | NONE       | (None)                        |
| 3            | PDF        | Acrobat (*.pdf)               |
| 4            | CSV        | Comma delimited (*.csv)       |
| 5            | HP         | HP Format (*.lis)             |
| 6            | HTM        | HTML Documents (*.htm)        |
| 7            | LP         | Line printer format (*.lis)   |
| 8            | WKS        | Lotus 1-2-3 files             |
| 9            | XLS        | Microsoft Excel Files (*.xls) |
| 10           | DOC        | Microsoft Word (*.doc)        |
| 11           | PS         | Postscript (*.lis)            |
| 12           | RPT        | Crystal Report (*.rpt)        |
| 13           | RTF        | Rich Text File (*.rtf)        |
| 14           | SPF        | SQR Portable Format (*.spf)   |
| 15           | TXT        | Text Files (*.txt)            |
| 16           | OTHER      | Other                         |
| 17           | Default    | Default                       |
| 18           | XML        | XML Format (*.xml)            |
| 19           | DAT        | Data Mover Data File (*.dat)  |

#### Notes:

- Your agent administrator can specify a default output format for all PeopleSoft jobs by setting the ps.default.outDestFormat parameter in the agent's agentparm.txt file.
- The ps\_dest\_format attribute overrides the default output format specified in the agent's agentparm.txt.

#### **Example: Format the Report Output as a Text File**

This example runs the PAYROLL process that has the Application Engine process type and PS\_ALL run control ID. The PAYROLL process output is formatted as a text file.

```
insert_job: ps_txfile
job_type: ps
machine: psagt
owner: VP1@server01
ps_output_dest:"c:\\temp\\payroll.txt"
ps_process_name: PAYROLL
ps_process_type: Application Engine
ps_dest_type: FILE
ps_dest_format: TXT
ps_run_ctrl_id: PS_ALL
```

## **ps\_dest\_type Attribute—Specify the Output Type for a PeopleSoft Report**

The ps\_dest\_type attribute specifies the output destination type for a PeopleSoft report.

**Note:** If you do not specify the ps\_dest\_type attribute in your job definition, the job uses the default output type specified in the agent's agentparm.txt file if it is defined.

#### **Supported Job Type**

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

## Syntax

This attribute has the following format:

`ps_dest_type: NONE | FILE | PRINTER | EMAIL | WEB`

### **NONE**

Specifies no output destination type.

### **FILE**

Sends the output of the PeopleSoft report to a file.

### **PRINTER**

Sends the output of the PeopleSoft report to a printer.

### **EMAIL**

Sends the output of the PeopleSoft report as an email message.

### **WEB**

Posts the output of the PeopleSoft report on a website.

### **Notes:**

- PeopleSoft stores the list of output destination types in the PSXLATITEM table.
- Your agent administrator can specify a default output type for all PeopleSoft jobs by setting the `ps.default.outDestType` parameter in the agent's `agentparm.txt` file.
- The `ps_dest_type` attribute overrides the default output type specified in the agent's `agentparm.txt`.
- If you specify EMAIL or WEB as the output destination type, you can distribute the PeopleSoft report electronically to operators, groups of people, or individuals.
- This attribute corresponds to the Type field in PeopleSoft.

### **Example: Format the Report Output as a Text File**

This example runs the PAYROLL process that has the Application Engine process type and PS\_ALL run control ID. The PAYROLL process output is formatted as a text file.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
owner: VP1@server01
ps_output_dest:"c:\\temp\\payroll.txt"
ps_process_name: PAYROLL
ps_process_type: Application Engine
ps_dest_type: FILE
ps_dest_format: TXT
ps_run_ctrl_id: PS_ALL
```

## ps\_detail\_folder Attribute—Specify the Name of a PeopleSoft Distribution Detail Folder

The ps\_detail\_folder attribute specifies the name of a report folder for the contents of a distribution list.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_detail\_folder: *folder\_name*  
***folder\_name***

Specifies the name of a distribution detail folder. This value corresponds to the Folder Name field in PeopleSoft.

**Limits:** Up to 18 characters; it cannot contain delimiters (such as spaces)

### Example: Specify a Distribution Detail Folder

This example runs the AEMINITEST process that has the process type, Application Engine. The output is sent to a website. The distribution details are stored in the GENERAL folder.

```
insert_job: ps_dsfolder
job_type: ps
machine: psagt
ps_process_name: AEMINITEST
ps_process_type: Application Engine
ps_dest_type: WEB
ps_dest_format: PDF
ps_run_ctrl_id: PS_ALL
owner: VP1@server01
ps_detail_folder: GENERAL
ps_server_name: PSPROD
```

## ps\_dlist\_roles Attribute—Specify a Distribution List of Roles

The ps\_dlist\_roles attribute specifies a distribution list of the roles that represent the individuals who are receiving the PeopleSoft report. For example, you can have a group of users defined as operators. Rather than specifying all the individual users to send the report, you can use this attribute to reference the role and let PeopleSoft handle it.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_dlist\_roles: *role*[,*role*...]

#### *role*

Specifies a role or a list of roles to send the report to. This value corresponds to the ID Type field (with Role selected) and the Distribution ID field in PeopleSoft.

**Limits:** Case-sensitive

#### Notes:

- You can specify multiple roles. Separate each role with a comma.
- The entire value can be up to 256 characters.
- An alternative to using this attribute is to use operator IDs as specified by the ps\_dlist\_users attribute.

### Example: Specify Multiple Roles

This example runs a Crystal report under the VP1 operator ID. The report output type is WEB and the output format is PDF. The report is distributed to the CLERK, BANK MANAGER, and Employee roles.

```
insert_job: ps_roles
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: WEB
ps_dest_format: PDF
ps_dlist_roles: CLERK,BANK MANAGER,Employee
owner: VP1@server01
```

#### **Example: Specify One Role**

This example runs a Crystal report under the VP1 operator ID. The report output type is WEB and the output format is PDF. The report is distributed to the BANK MANAGER role.

```
insert_job: ps_roles
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: WEB
ps_dest_format: PDF
ps_dlist_roles: BANK MANAGER
owner: VP1@server01
```

## **ps\_dlist\_users Attribute—Specify a Distribution List of Operator IDs**

The ps\_dlist\_users attribute specifies a distribution list of operator IDs to send a PeopleSoft report to.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

#### **Supported Job Type**

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

#### **Syntax**

This attribute has the following format:

`ps_dlist_users: operator_id[,operator_id...]`

#### ***operator\_id***

Specifies an operator ID to send the report to. This value corresponds to the ID Type field (with User selected) and the Distribution ID field in PeopleSoft.

**Limits:** Case-sensitive

**Default:** The operator ID specified in the owner attribute

**Notes:**

- You can specify multiple operator IDs. Separate each operator ID with a comma.
- The entire value can be up to 256 characters.
- By default, the report output is sent to the operator ID specified in the owner attribute. If you do not want to distribute the output to any operator IDs, you must include a ps\_dlist\_users attribute in the job definition and set it to two single quotes or provide no value.
- Instead of specifying a list of operator IDs to distribute the report output to, you can distribute the report output to a list of roles using the ps\_dlist\_roles attribute.
- If the output destination type is EMAIL or WEB, you can also distribute the report output to a list of email addresses using the ps\_email\_address attribute.

**Example: Specify Multiple Operator IDs**

This example runs a Crystal report under the VP3 operator ID. The report is formatted as PDF and distributed in an email to the VP1, VP2, and VP3 operator IDs.

```
insert_job: ps_users
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: EMAIL
ps_dest_format: PDF
ps_dlist_users: VP1,VP2,VP3
ps_operator_id: VP3@ps1
ps_run_ctrl_id: test
```

**Example: Distribute a Report to Email Addresses Instead of a Distribution List**

This example runs a Crystal report under the VP2 operator ID. The output is stored as a PDF web report. The ps\_dlist\_users attribute has no value, so the output is sent to the email addresses specified in the ps\_email\_address attribute. The email includes a subject title and body text. The web report and the job logs are included with the email.

```
insert_job: ps_nousers
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: WEB
ps_dest_format: PDF
ps_dlist_users:
owner: VP2@server01
ps_email_web_report: YES
ps_email_address: user1@example.com;user2@example.com
ps_email_subject: PeopleSoft Report Status
ps_email_text: This report is available for distribution.
ps_email_log: YES
```

## ps\_email\_address Attribute—Specify the Email Addresses on a Distribution List

The ps\_email\_address attribute specifies the email addresses of the recipients on a distribution list. The PeopleSoft report is emailed to the recipients.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_email\_address: *address*[;*address*...]

#### *address*

Specifies an email address on a distribution list to send the report to. This value corresponds to the Email Address List field in PeopleSoft.

**Limits:** Case-sensitive

#### Notes:

- You can specify multiple email addresses. Separate each address with a semi-colon.
- The entire value can be up to 256 characters.

### Example: Send a PeopleSoft Report to Two Email Addresses

This example runs a Crystal report and emails the output to recipients. The Crystal report runs under the VP2 operator ID. The output is sent to the email addresses specified in the ps\_email\_address attribute. The email includes a subject title.

```
insert_job: ps_email
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: EMAIL
ps_dest_format: PDF
ps_operator_id: VP2@ps1
ps_email_address: user1@example.com;user2@example.com
ps_email_subject: PeopleSoft Report Status
ps_email_text: This report is available for distribution.
```

## ps\_email\_address\_expanded Attribute—Specify Additional Email Addresses on a Distribution List

The ps\_email\_address\_expanded attribute lets you specify additional email addresses for the recipients on a distribution list. When the email addresses you specify in the ps\_email\_address attribute exceeds the limit, you can use this attribute to extend the email address list.

**Note:** To use this attribute, you must specify the ps\_email\_address attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_email_address_expanded: address[;address...]`

#### *address*

Specifies an email address on a distribution list to send the report to. This value corresponds to the Email Address List field in PeopleSoft.

**Limits:** Case-sensitive

#### Notes:

- You can specify multiple email addresses. Separate each address with a semi-colon.
- The entire value can be up to 256 characters.

## ps\_email\_log Attribute—Specify Whether to Email PeopleSoft Job Logs

The ps\_email\_log attribute specifies whether to email job logs along with the PeopleSoft report to recipients on a distribution list.

### Notes:

- To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute. You must also specify the ps\_email\_address attribute in the job definition.
- If you do not specify the ps\_email\_log attribute in your job definition, job logs are not emailed to recipients (the default).

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_email\_log: YES | NO

#### YES

Emails job logs along with the report to the recipients on the distribution list.

#### NO

Does not email job logs to the recipients on the distribution list. This is the default.

**Note:** This attribute corresponds to the Email With Log field in PeopleSoft.

### Example: Email PeopleSoft Job Logs

This example emails the report in PDF format and the PeopleSoft job logs to the VP1 user. The email includes a subject title and body text.

```
insert_job: ps_logs
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: EMAIL
ps_dest_format: PDF
ps_dlist_users: VP1
owner: VP2@server01
ps_email_subject: PeopleSoft Report Logs
ps_email_text: This report is available for distribution.
ps_email_log: YES
```

## ps\_email\_subject Attribute—Define an Email Subject

The ps\_email\_subject attribute defines an email subject to include in the email to distribute to recipients of a PeopleSoft report.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_email_subject: subject_text`

#### *subject\_text*

Defines an email subject to include in the email. This value corresponds to the Email Subject field in PeopleSoft.

**Limits:** Up to 256 characters; case-sensitive

### Example: Include an Email Subject in an Email

This example runs a Crystal report and emails the output to recipients. The Crystal report runs under the VP2 operator ID. The output is sent to the email addresses specified in the ps\_email\_address attribute. The email includes a subject title.

```
insert_job: ps_email
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: EMAIL
ps_dest_format: PDF
ps_operator_id: VP2@ps1
ps_email_address: user1@example.com;user2@example.com
ps_email_subject: PeopleSoft Report Status
ps_email_text: This report is available for distribution.
```

## ps\_email\_text Attribute—Define the Body Text of an Email

The ps\_email\_text attribute defines the body text of the email to distribute to recipients of a PeopleSoft report.

**Note:** To use this attribute, you must specify EMAIL or WEB for the ps\_dest\_type attribute.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_email\_text: *email\_text*

#### *email\_text*

Defines the body text of the email. This value corresponds to the Message Text field in PeopleSoft.

**Limits:** Up to 1000 characters; case-sensitive

#### **Example: Include Body Text in an Email**

This example runs a Crystal report and emails the output to recipients. The Crystal report runs under the VP2 operator ID. The output is sent to the email addresses specified in the ps\_email\_address attribute. The email includes a subject title.

```
insert_job: ps_email
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: EMAIL
ps_dest_format: PDF
ps_operator_id: VP2@ps1
ps_email_address: user1@example.com;user2@example.com
ps_email_subject: PeopleSoft Report Status
ps_email_text: This report is available for distribution.
```

## **ps\_email\_web\_report Attribute—Specify Whether to Email a PeopleSoft Web Report**

The ps\_email\_web\_report attribute specifies whether to email a web report to recipients on a distribution list.

#### **Notes:**

- To use this attribute, you must specify WEB for the ps\_dest\_type attribute. You must also specify one or more addresses using the ps\_email\_address attribute.
- If you do not specify the ps\_email\_web\_report attribute in your job definition, a web report is not emailed to recipients (the default).

#### **Supported Job Type**

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

## Syntax

This attribute has the following format:

ps\_email\_web\_report: YES | NO

**YES**

Emails a web report to the recipients on the distribution list.

**NO**

Does not email a web report to the recipients on the distribution list. This is the default.

**Note:** This attribute corresponds to the Email Web Report field in PeopleSoft.

### Example: Email a PeopleSoft Web Report

This example stores the output as a PDF web report. The web report is sent to the email addresses specified in the ps\_email\_address attribute. The email includes a subject title and body text.

```
insert_job: ps_web
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Crystal
ps_dest_type: WEB
ps_dest_format: PDF
ps_operator_id: VP2@ps1
ps_email_web_report: YES
ps_email_address: user1@example.com;user2@example.com
ps_email_subject: PeopleSoft Report Status
ps_email_text: This report is available for distribution.
```

## ps\_operator\_id Attribute—Specify a PeopleSoft Operator ID

The ps\_operator\_id attribute specifies the operator ID under whose authority the PeopleSoft reports run. The value you specify for this attribute overrides the default defined on the agent by the ps.default.oprId parameter in the agentparm.txt file.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

**Note:** A PeopleSoft job must have a valid operator ID to run successfully. Check with your agent administrator to determine whether a default operator ID is set on the agent. If a default is not set, then you must specify the ps\_operator\_id attribute in a job definition.

### Syntax

This attribute has the following format:

`ps_operator_id: psopr@tag`

**`psopr@tag`**

Specifies a valid operator ID under whose authority the PeopleSoft reports run, where *psopr* is the PeopleSoft operator ID and the entire string is used to look up the password that is stored using `autosys_secure`.

**Limits:** Up to 30 characters total; *psopr* is restricted to five characters

**Example:** `ptdmo@peoplesoft`

#### Notes:

- The `ps_operator_id` must be defined on CA Workload Automation AE using the `autosys_secure` command unless the `ps.skipOprPswdValidation=true` parameter is configured in the `agentparm.txt` file.
- You can omit the `ps_operator_id` if the default operator ID, specified by the `ps.default.oprId` parameter in the `agentparm.txt`, applies for the job.

## ps\_output\_dest Attribute—Specify an Output Path for a PeopleSoft Job

The `ps_output_dest` attribute specifies the output destination for the PeopleSoft request. The destination can be a file directory or a printer. The value you specify for this attribute overrides the default destination defined on the PeopleSoft system.

**Note:** To use this attribute, you must specify FILE or PRINTER for the `ps_dest_type` attribute.

### Supported Job Type

This attribute is required for the [PeopleSoft \(PS\) job type](#) (see page 249) if FILE or PRINTER is specified for the `ps_dest_type` attribute.

## Syntax

This attribute has the following format:

`ps_output_dest: file_path | printer_path`

### *file\_path*

Specifies the path to the output directory and the output file name.

**Limits:** Up to 127 characters; case-sensitive

**Note:** Specify this path when the ps\_dest\_type is FILE.

### *printer\_path*

Specifies the network location of the printer including the printer server and shared printer name.

**Limits:** Up to 127 characters; case-sensitive

#### **Notes:**

- Specify this path when the ps\_dest\_type is PRINTER.
- The specified printer must be set up on the PeopleSoft system.

**Note:** This value corresponds to the Output Destination field in PeopleSoft.

## Example: Direct the Output of a PeopleSoft Job to a File

This example runs an SQR Report. The report is formatted as PDF and outputted to a file.

```
insert_job: ps_file
job_type: ps
machine: psagent
ps_process_name: XRFWIN
ps_process_type: SQR Report
ps_dest_type: FILE
ps_dest_format: PDF
ps_output_dest: /export/home/PSoutput/report1.pdf
ps_run_ctrl_id: test
owner: VP1@ps1
```

**Example: Direct the Output of a PeopleSoft Job to a Printer**

This example runs an SQR Report. The report is formatted as PS and outputted to a printer.

```
insert_job: ps_printer
job_type: ps
machine: psagent
ps_process_name: XRFWIN
ps_process_type: SQR Report
ps_dest_type: PRINTER
ps_dest_format: PS
ps_output_dest: \\printers\PRINTER1
ps_run_ctrl_id: test
ps_operator_id: VP1@ps1
```

## ps\_process\_name Attribute—Specify a PeopleSoft Process to Run

The ps\_process\_name attribute specifies the name of the PeopleSoft process that the job runs.

### Supported Job Type

This attribute is required for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_process_name: process_name`

#### *process\_name*

Specifies the name of the PeopleSoft report to run. This value corresponds to the Process Name field in PeopleSoft. PeopleSoft stores the list of process names in the PS\_PRCDEFN table.

**Limits:** Up to 12 characters; case-sensitive

### Example: Run a PeopleSoft Process

This example runs the process named AEMINTEST. The process has the Application Engine type and PS\_ALL run control ID.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
ps_process_name: AEMINTEST
ps_process_type: Application Engine
ps_run_ctrl_id: PS_ALL
```

## ps\_process\_type Attribute—Specify the Type of PeopleSoft Process to Run

The ps\_process\_type attribute specifies the type of the PeopleSoft process that the job runs.

### Supported Job Type

This attribute is required for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_process_type: process_type`

#### *process\_type*

Specifies the PeopleSoft process type corresponding to the process that the job runs. PeopleSoft stores the list of process types in the PS\_PRCSTYPEDEFN table. This value corresponds to the Process Type field in PeopleSoft.

**Limits:** Up to 30 characters; case-sensitive

#### Examples:

- Application Engine
- COBOL SQL
- Crw Online
- Crystal
- Crystal Check
- Cube Builder
- Database Agent
- Demand Planning Upload
- Essbase
- HyperCube Builder
- Message Agent API
- nVision
- nVision-Report
- Optimization Engine
- SQR PO-Special Process
- SQR Process

- SQR Report
- SQR Report For WF Delivery
- Winword

#### **Example: Run an Application Engine PeopleSoft Process**

This example runs the process named AEMINTEST. The process has the Application Engine type and PS\_ALL run control ID.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
ps_process_name: AEMINTEST
ps_process_type: Application Engine
ps_run_ctrl_id: PS_ALL
```

## **ps\_restarts Attribute—Specify Whether to Disable the Restart of Failed PeopleSoft Jobs**

The ps\_restarts attribute specifies whether to disable a restart feature for previously failed jobs from the point where the job failed.

**Note:** If you do not specify the ps\_restart attribute in your job definition, the restart feature is enabled (the default). When a previously failed job is resubmitted, it restarts from where it stopped.

#### **Supported Job Type**

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

#### **Syntax**

This attribute has the following format:

ps\_restart: YES | NO

##### **NO**

Disables the restart feature. Previously-failed jobs restart from the beginning.

##### **YES**

Does not disable the restart feature. When a failed job is resubmitted, it restarts from where it stopped. This is the default.

**Note:** The restart feature only applies to certain PeopleSoft Process Types such as Application Engine.

#### Example: Disable the Restart of Failed PeopleSoft Jobs

This example disables the restart feature. Previously failed jobs restart from the beginning.

```
insert_job: ps_norestart
job_type: PS
machine: psagt
ps_process_name: XRFWIN
ps_process_type: Application Engine
ps_restarts: Yes
```

## ps\_run\_cntrl\_args Attribute—Specify the Arguments for a PeopleSoft Run Control Table

The `ps_run_cntrl_args` attribute specifies the run control arguments for a run control table on the PeopleSoft system. You can use this attribute with the `ps_run_control_table` attribute to add parameters to the run control table.

#### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

#### Syntax

This attribute has the following format:

`ps_run_cntrl_args: argument[, argument...]`

#### *argument*

Specifies a run control argument to add to the run control table.

**Limits:** Case-sensitive; each argument can be up to 1024 characters

**Note:** Enclose values that contain delimiters (such as spaces) or special characters in quotation marks.

**Examples:** BI\_WF\_0001, 0, "third arg"

**Notes:**

- You can specify multiple arguments. Separate each argument with a comma.
- You can specify multiple entries of ps\_run\_cntrl\_args to define different sets of arguments. For example, you can specify two sets of arguments using two ps\_run\_cntrl\_args attributes as follows:

```
ps_run_cntrl_args : BI_WF_0001, 0, 0, N, 0, N, , "b\", sysdate
ps_run_cntrl_args : BI_WF_0002, 0, 0, N, 0, N, , "b\", sysdate
```
- Each entry of ps\_run\_cntrl\_args can be up to 4096 characters.
- If the number of values specified in the ps\_run\_cntrl\_args attribute is less than the number of columns in the corresponding run control table, the agent populates the remaining columns with default values. The default value for strings is a space. The default value for all other data types is null.
- To indicate default values will be used, you can optionally add a comma at the end of the attribute, for example:

```
ps_run_cntrl_args: BI_WF_0001,0,0,N,0,N,
```
- If an argument cannot be updated with a default value, the job fails with an exception. For example, the job can fail if the database does not allow a null value within the table.
- For Oracle, if you specify date values other than SYSDATE in the job definition, you must use the following format:

"DD-MM-YYYY HH24:MI:SS"

Oracle date values are processed using the to\_date() function. For example, for the argument value 26-07-2009, the agent invokes to\_date('26-07-2009','DD-MM-YYYY HH24:MI:SS') and "2009-07-26 00:00:00.0" is inserted into the run control table.

**Example: Add a Row of Process Parameters to a Run Control Table**

In this example, the `ps_run_cntrl_table` attribute specifies the name of the run control table that contains the run control parameters for the XYZ process. The job inserts a new row in the `PS_AEREQUESTTBL` run control table with data from the run control arguments (`ps_run_cntrl_args`).

```
insert_job: ps_params
job_type: ps
machine: psagent
ps_process_name: XYZ
ps_process_type: Application Engine
owner: VP1@server01
ps_run_cntrl_id: TEST
ps_run_control_table: PS_AEREQUESTTBL
ps_run_cntrl_args: BI_WF_0001, 0, 0, N, 0, N, , "b\", sysdate
```

The columns in the `PS_AEREQUESTTBL` run control table correspond to the run control arguments as follows:

| Column            | Data type    | Value               |
|-------------------|--------------|---------------------|
| AE_APPLID         | VARCHAR2(12) | BI_WF_0001          |
| CURTEMPINSTANCE   | NUMBER(38)   | 0                   |
| PROCESS_FREQUENCY | VARCHAR2(1)  | O                   |
| AE_PROCESS_STATUS | VARCHAR2(1)  | N                   |
| PROCESS_INSTANCE  | NUMBER(10)   | 0                   |
| PROCESS_ORIG      | VARCHAR2(1)  | N                   |
| LAST_RUN_DTM      | DATE         | null                |
| MARKET            | VARCHAR2(3)  | b\                  |
| ASOF_DT           | DATE         | 2010-07-10 14:30:00 |

## ps\_run\_cntrl\_id Attribute—Specify PeopleSoft Run Parameters

The ps\_run\_cntrl\_id attribute specifies a set of PeopleSoft run parameters for a PeopleSoft process.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_run\_cntrl\_id: *control\_id*

#### *control\_id*

Specifies the value assigned to the run control identifier. This value corresponds to the Run Control ID field in PeopleSoft.

**Limits:** Up to 30 characters; case-sensitive

**Example:** FLOOR8\_COLOR

**Note:** In PeopleSoft, some processes spawn children using the CreateProcessRequest PeopleCode function. PeopleSoft may mark the parent process as a successful completion even if one or more of its children fail. If you want the agent to check the status of children when determining the completion status of the parent, add ".\*" to the end of the Run Control ID, for example, PS\_ALL.\*.

### Notes:

- If you do not specify this attribute in the job definition, a default run control ID must be defined in the ps.default.runCntlId parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- The ps\_run\_cntrl\_id attribute overrides the default run control ID specified in the agent's agentparm.txt file.

### **Example: Specify a Run Control ID**

This example runs the process named AEMINTEST. The process has the Application Engine type and PS\_ALL run control ID.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
ps_process_name: AEMINTEST
ps_process_type: Application Engine
ps_run_cntrl_id: PS_ALL
```

**Example: Check Children Status**

This example runs the AEMINITEST process on the agent named psagt. The job completes successfully if the AEMINITEST process and all of its children complete successfully.

```
insert_job: ps_children
job_type: ps
machine: psagt
ps_process_name: AEMINITEST
ps_process_type: Application Engine
owner: VP1@server01
ps_run_ctrl_id: HDL8010.*
```

## **ps\_run\_control\_table Attribute—Specify the Table that Contains the PeopleSoft Run Parameters**

The `ps_run_control_table` attribute specifies the name of the table that contains the run control parameters for a PeopleSoft process. You can use this statement with the `ps_run_ctrl_args` attribute to add parameters to the run control table.

**Supported Job Type**

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

**Syntax**

This attribute has the following format:

```
ps_run_control_table: control_table
control_table
```

Specifies the name of the table that contains the run parameters for a given PeopleSoft process.

**Limits:** Up to 50 characters; case-sensitive

**Example:** PS\_AEREQUESTTBL

**Example: Add a Row of Process Parameters to a Run Control Table**

In this example, the `ps_run_cntrl_table` attribute specifies the name of the run control table that contains the run control parameters for the XYZ process. The job inserts a new row in the `PS_AEREQUESTTBL` run control table with data from the run control arguments (`ps_run_cntrl_args`).

```
insert_job: ps_params
job_type: ps
machine: psagent
ps_process_name: XYZ
ps_process_type: Application Engine
owner: VP1@server01
ps_run_ctrl_id: TEST
ps_run_control_table: PS_AEREQUESTTBL
ps_run_ctrl_args: BI_WF_0001, 0, 0, N, 0, N, , "b\", sysdate
```

The columns in the `PS_AEREQUESTTBL` run control table correspond to the run control arguments as follows:

| Column            | Data type    | Value               |
|-------------------|--------------|---------------------|
| AE_APPLID         | VARCHAR2(12) | BI_WF_0001          |
| CURTEMPINSTANCE   | NUMBER(38)   | 0                   |
| PROCESS_FREQUENCY | VARCHAR2(1)  | O                   |
| AE_PROCESS_STATUS | VARCHAR2(1)  | N                   |
| PROCESS_INSTANCE  | NUMBER(10)   | 0                   |
| PROCESS_ORIG      | VARCHAR2(1)  | N                   |
| LAST_RUN_DTM      | DATE         | null                |
| MARKET            | VARCHAR2(3)  | b\                  |
| ASOF_DT           | DATE         | 2010-07-10 14:30:00 |

## ps\_server\_name Attribute—Specify a Server to Run the PeopleSoft Job

The ps\_server\_name attribute specifies the name of the target server that runs the PeopleSoft job.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_server_name: server_name`

#### *server\_name*

Specifies the name of the target server that runs the PeopleSoft job. PeopleSoft stores the list of server names in the PS\_SERVERDEFN table. This value corresponds to the Server Name field in PeopleSoft.

**Limits:** Up to 8 characters; cannot contain delimiters (such as spaces)

**Example:** PSNT

### Notes:

- Your agent administrator can specify a default run server name for all PeopleSoft jobs by setting the ps.default.serverName parameter in the agent's agentparm.txt file.
- The ps\_server\_name attribute overrides the default run server name specified in the agent's agentparm.txt.
- If the server name is not defined on the agent or using the ps\_server\_name attribute, the PeopleSoft Scheduler determines the server that will run the job.

### Example: Specify a Server to Run a PeopleSoft Process

This example runs a process named DDDAUDIT. The server named PSPR runs the job. The job runs in Central time in the Continental United States.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
ps_process_name: DDDAUDIT
ps_process_type: SQR Report
ps_server_name: PSPR
ps_time_zone: CST
```

## ps\_skip\_parm\_updates Attribute—Specify Whether to Update Job Parameters

The ps\_skip\_parm\_updates attribute specifies whether the agent updates job parameters with data in the PS\_PRCDEFN table.

### Notes:

- If you do not specify the ps\_skip\_parm\_updates attribute in your job definition, the agent updates job parameters with data in the PS\_PRCDEFN table (the default).
- We recommend that you skip parameter updates when some bind variables in the PS\_PRCDEFN table may not be suitably defined.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

ps\_skip\_parm\_updates: YES | NO

#### YES

Specifies that the agent does not update job parameters with data in the PS\_PRCDEFN table. If you specify YES, you can use the ps\_args attribute to pass additional argument values.

#### NO

Specifies that the agent updates job parameters with data in the PS\_PRCDEFN table. This is the default.

#### **Example: Run a PeopleSoft Process Without Updating the PS\_PRCDEFN Table**

This example runs the OMC3220- process on the agent named psagt. The job parameters are not updated with data in the PS\_PRCDEFN table. The ps\_run\_cntrl\_args attribute passes additional parameters for the report.

```
insert_job: ps_skipparms
job_type: ps
machine: psagt
ps_process_name: OMC3220-
ps_process_type: Crystal
ps_dest_type: WEB
ps_dest_format: PDF
ps_run_cntrl_id: test
owner: VP1@server01
ps_skip_parm_updates: YES
ps_args: -TEST, SHARE
```

#### **Example: Ignore Parameters in the PS\_PRCDEFN Table and Pass Additional Parameters to the nVision Report**

In this example, the job parameters are not updated with data in the PS\_PRCDEFN table. The ps\_run\_cntrl\_args attribute passes additional parameters for the nVision report.

```
insert_job: ps_skipparms2
job_type: ps
machine: psagt
ps_process_name: NVSRUN
ps_process_type: nVision-Report
ps_dest_type: FILE
ps_dest_format: XLS
ps_output_dest: "c:\test\testnv.xls"
ps_skip_parm_updates: YES
ps_args: "-NRNVARIBLE -NBUAUS01 -NHLhttp://10.1.1.40/psp/ps
/EMPLOYEE/ERP/c/REPORT_BOOKS.IC_RUN_DRILLDOWN.GBL?Action=A"
```

#### **Example: Update Job Parameters with Data in the PS\_PRCDEFN Table**

This example updates job parameters with data in the PS\_PRCDEFN table.

```
insert_job: ps_updateparms
job_type: ps
machine: psagt
ps_process_name: OMC3220-
ps_process_type: Crystal
ps_run_cntrl_id: sample
ps_skip_parm_updates: NO
owner: VP2@server01
```

## ps\_time\_zone Attribute—Specify a Time Zone

The ps\_time\_zone attribute specifies a different time zone for the PeopleSoft report being run.

### Supported Job Type

This attribute is optional for the [PeopleSoft \(PS\) job type](#) (see page 249).

### Syntax

This attribute has the following format:

`ps_time_zone: time_zone`

#### *time\_zone*

Specifies a different time zone for the report being run. This value corresponds to the Time Zone field in PeopleSoft.

**Limits:** Up to 9 characters; cannot contain delimiters (such as spaces)

**Note:** You can view the time zone settings in PeopleSoft.

### Example: Specify a Time Zone

This example runs a process named DDDAUDIT. The server named PSPR runs the job. The job runs in Central time in the Continental United States.

```
insert_job: ps_txtfile
job_type: ps
machine: psagt
ps_process_name: DDDAUDIT
ps_process_type: SQR Report
ps_server_name: PSPR
ps_time_zone: CST
```

## remote\_name Attribute—Specify a Remote Class Name

The remote\_name attribute specifies the naming class of the Java object you want to invoke a method on in an RMI job.

### Supported Job Type

This attribute is required for the [RMI \(JAVARMI\) job type](#) (see page 254).

### Syntax

This attribute has the following format:

`remote_name: "remote_class"`  
`"remote_class"`

Specifies the reference location of the object you want to invoke a method on. The format is the following:

`"rmi://[host:port/]name"`

**host**

(Optional) Specifies the name of the local or remote computer where the RMI registry that hosts the remote object runs.

**Default:** local host

**port**

(Optional) Specifies the RMI registry port number the host uses to listen for requests.

**Default:** 1099, the default port of the host's RMI registry

**name**

Specifies the name of a remote object.

**Limits:** Up to 256 characters; case-sensitive

**Example:** "rmi://smith:5000/test"

### Example: Invoke a Method to Start a Remote Server

Suppose that you want to invoke a method that starts a remote server using remote object activation. You want the server to start immediately.

```
insert_job: start
job_type: JAVARMI
machine: appagent
remote_name: "rmi://remotehost/Test"
method_name: startserver
j2ee_parameter: String="now"
```

## request\_id Attribute—Specify Request ID of the Oracle E-Business Suite Request to Copy

The request\_id attribute specifies the request ID of the Oracle E-Business Suite request you want to copy. You can copy an existing single request defined on Oracle E-Business Suite and run it under the agent.

### Supported Job Type

This attribute is required for the [Oracle E-Business Suite Copy Single Request \(OACOPY\) job type](#) (see page 242).

### Syntax

This attribute has the following format:

request\_id: *request\_id*

#### *request\_id*

Specifies the request ID of the single request you want to copy. In Oracle Applications, the request ID is found in the Request ID column of the Requests dialog.

**Limits:** Up to 20 numeric digits

**Example:** 49389545970965675096795

### Example: Copy a Single Request

Suppose that you want to copy an existing single request defined on Oracle E-Business Suite. In this example, the job copies the single request job with request ID 2255470 and overrides the Oracle E-Business Suite user name and responsibility name defined in the agentparm.txt file.

```
insert_job: oacopy_single
job_type: oacopy
machine: oaagent
request_id: 2255470
oracle_user: SYSADMIN
oracle_resp: System Administrator
```

## result\_type Attribute—Specify the Data Type to be Returned from the Stored Procedure

The result\_type attribute specifies the data type to be returned from the stored procedure. If the type returned from the stored procedure and the value specified for the result\_type attribute do not match, the job fails.

### Supported Job Type

This attribute is optional for the [Database Stored Procedure \(DBPROC\) job type](#) (see page 209).

### Syntax

This attribute has the following format:

`result_type: data_type`

#### *data\_type*

Specifies the data type to be returned from the stored procedure.

The following table lists the supported data types for different databases. The data types listed in the database columns are defined in the procedure definition within the database. In the job definition, use the corresponding JDBC data type from the first column.

| JDBC Data Type | Valid Input Format | Oracle                    | Microsoft SQL Server | DB2           |
|----------------|--------------------|---------------------------|----------------------|---------------|
| CHAR           | Plain text         | CHAR                      | CHAR                 | CHAR          |
| VARCHAR        | Plain text         | VARCHAR2(x)<br>VARCHAR(x) | VARCHAR(x)           | VARCHAR(x)    |
| LONGVARCHAR    |                    | Not supported             | Not supported        | Not supported |
| NUMERIC        | Number             | NUMBER,<br>NUMERIC        | NUMERIC              | NUMERIC       |
| DECIMAL        | Number             | DECIMAL                   | DECIMAL              | DECIMAL       |
| BIT            |                    | Not supported             | BIT                  | Not supported |
| BOOLEAN        |                    | Not supported             | Not supported        | Not supported |
| TINYINT        |                    | Not supported             | TINYINT              | Not supported |
| SMALLINT       | Number             | SMALLINT                  | SMALLINT             | SMALLINT      |
| INTEGER        | Number             | INTEGER                   | INTEGER              | INTEGER       |

| JDBC Data Type | Valid Input Format                     | Oracle        | Microsoft SQL Server | DB2       |
|----------------|----------------------------------------|---------------|----------------------|-----------|
| BIGINT         | Number                                 | Not supported | BIGINT               | BIGINT    |
| REAL           | Number<br>[dot Number]                 | REAL          | REAL                 | REAL      |
| FLOAT          | Number<br>[dot Number]                 | FLOAT         | FLOAT                | FLOAT     |
| DOUBLE         |                                        |               |                      | DOUBLE    |
| DATE           | yyyy-mm-dd                             | DATE          |                      | DATE      |
| DATE           | String (for example,<br>SYSDATE)       |               |                      |           |
| TIME           | hh:mm:ss                               |               |                      | TIME      |
| TIMESTAMP      | yyyy-mm-dd<br>hh:mm:ss.fffffffff<br>ff | TIMESTAMP     |                      | TIMESTAMP |

## return\_class\_name Attribute—Specify a Return Class Name

The return\_class\_name attribute specifies the Java class name of the return value in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

`return_class_name: return_class`

***return\_class***

Specifies the Java class name of the return value.

**Limits:** Up to 1024 characters; case-sensitive

### Example: Specify Java Class Name of Stock Quote

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webservicex.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type string in the return namespace `http://www.webservicex.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

## return\_namespace Attribute—Specify an XML Namespace

The `return_namespace` attribute specifies the XML namespace for the `return_xml_name` attribute in a Web Service job. If you omit this attribute, the return namespace defaults to the target namespace specified by the `target_namespace` attribute.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

```
return_namespace: return_namespace
```

***return\_namespace***

Specifies the XML namespace for the XML type that maps to the Java class name of the return value.

**Limits:** Up to 256 characters; case-sensitive

**Example: Specify Return Namespace for the XML Type of Stock Quote**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webservicex.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type `string` in the return namespace `http://www.webservicex.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

## return\_xml\_name Attribute—Specify an XML Type

The return\_xml\_name attribute specifies the XML type that maps to the return\_class\_name attribute in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

**Note:** This attribute is required if the return\_class\_name attribute is specified in the job definition.

### Syntax

This attribute has the following format:

return\_xml\_name: *XML\_type*

#### *XML\_type*

Specifies the XML type that maps to the Java class name of the return value.

**Limits:** Up to 256 characters; case-sensitive

### Example: Specify XML Type of Stock Quote

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webserviceX.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type string in the return namespace `http://www.webserviceX.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## run\_calendar Attribute—Identify a Custom Calendar

The run\_calendar attribute identifies a custom calendar to use to determine the days of the week on which to run the job you are defining. You can use a standard calendar or an extended calendar as the run calendar.

Custom calendars are useful for complex date specification, such as when you need to run a job on the last business day of the month. Custom calendars can specify any number of dates on which to run associated jobs. Each calendar is stored as a separate uniquely named object in the database and can be associated with one or more jobs. The specified calendar must have previously been created using the autocal\_asc command.

CA Workload Automation AE schedules the job to run on every day defined in the specified calendar, at the times defined in the calendar or in the start\_times or start\_mins attributes.

**Note:** Times set in the start\_times and start\_mins attributes override times set in a custom calendar. The date\_conditions attribute controls the usage of this attribute.

You cannot specify both the days\_of\_week and run\_calendar attributes in the same job definition. If you do, CA Workload Automation AE ignores both attributes and generates an error.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`run_calendar: calendar_name`

#### *calendar\_name*

Defines the name of a custom calendar that was created with the autocal\_asc command.

**Default:** No calendar is used.

**Limits:** Up to 30 alphanumeric characters

### Example: Specify a Job to Run on Days Specified in the Previously Created Custom Calendar

This example specifies that the job should run on the days specified in the previously created custom calendar last\_business:

```
date_conditions: y
run_calendar: last_business
```

## run\_window Attribute—Define an Interval for a Job to Start

The run\_window attribute defines an interval during which the job you are defining may start. This attribute controls only when the job starts—not when it stops.

When a job definition contains the run\_window attribute and the job becomes eligible to run (based on its starting conditions), CA Workload Automation AE verifies whether the specified run window includes the current time. If it does, the job starts. If it does not, the product determines whether to run the job based on the end of the previous run window and the beginning of the next run window.

- When the current time is closer to the beginning of the next run window, the product schedules the job to start when the next run window starts.
- When the current time is closer to the end of the previous run window, the product does not start the job and changes its status to INACTIVE.

The run\_window attribute is not in itself a starting condition—it is an additional control over when a job may start after its starting conditions are satisfied. A run window cannot span more than 24 hours.

This attribute is especially useful, for example, when you do not know when a monitored file may arrive and there are specific times when a job dependent on the monitored file should not run.

**Note:** Jobs must have starting conditions in addition to the run\_window attribute for the job to start automatically. The date\_conditions attribute controls the usage of this attribute.

If the job is in a box, its status changes to ACTIVATED when the box starts running. However, if the current time is not in the specified run window for the job, its status changes to INACTIVE.

- When the current time is closer to the end of the previous run\_window, the job's status changes to INACTIVE. The box job can still run to completion.
- When the current time is closer to the beginning of the next run\_window, the product issues a future STARTJOB event for the job for the next run\_window.

These actions enable a box job to run to completion when the run\_window for a job in the box has just closed.

For example, assume JobA, JobB, and JobC are in Box1, and JobA has a run\_window of 02:00 to 04:00 and another starting condition such as start\_mins or start\_times. If Box1 starts at 04:05, JobB and JobC can run and JobA becomes INACTIVE so that the box can complete that day. If Box1 instead starts at 16:05, JobA will have a STARTJOB event set for 02:00 the next day, and the box continues running until the job starts the next day.

You can use the sendevent command to define times of the day when you do not want the job to start by putting the job into JOB\_ON\_HOLD status and then changing its status to JOB\_OFF\_HOLD later.

### **Supported Job Types**

This attribute is optional for all job types.

### **Syntax**

This attribute has the following format:

`run_window: "time-time"`

#### ***time-time***

Defines the interval during which a job may start in the format "*hh:mm-hh:mm*", where *hh* denotes hours (in 24-hour format) and *mm* denotes minutes. You must enclose the interval in quotation marks ("") and separate the beginning and ending times with a hyphen (-). The specified interval can overlap midnight, but cannot encompass more than 24 hours.

**Limits:** Up to 20 alphanumeric characters

**Default:** No run window is used.

#### Example: Specify an Interval for the Job to Start

This example specifies that the job may start only between 11:00 p.m. and 2:00 a.m., regardless of other conditions.

```
date_conditions: y
run_window: "23:00-02:00"
start_mins: "10,20"
```

## sap\_abap\_name Attribute—Run ABAP Steps Within an SAP Job

The sap\_abap\_name attribute identifies the ABAP step to be run.

#### Supported Job Type

This attribute is optional for the [SAP Process Monitor \(SAPPM\) job type](#) (see page 264) when sap\_process\_status is set to RUNNING or STOPPED.

**Note:** You cannot use this attribute with sap\_process\_status set to WAITING.

#### Syntax

This attribute has the following format:

```
sap_abap_name: abap
abap
```

Specifies the valid SAP system ABAP name. The ABAP name corresponds to the SAPGUI ABAP program Name field on the Create Step dialog.

**Limits:** Up to 80 valid SAP characters (for SAP version 4.5/4.6)

#### Example: Run an ABAP Step

This example defines a job that runs the RSPARAM ABAP.

```
insert_job: SAPPM_JOB
job_type: SAPPM
machine: SAPAGENT
sap_process_status: RUNNING
sap_abap_name: RSPARAM
```

## sap\_chain\_id Attribute—Identify Name of Business Warehouse Process Chain on SAP

The sap\_chain\_id attribute identifies the name of a Business Warehouse Process Chain on the SAP system.

### Supported Job Type

This attribute is required for the [SAP BW Process Chain \(SAPBWPC\) job type](#) (see page 258).

### Syntax

This attribute has the following format:

`sap_chain_id: process_chain`

`process_chain`

Specifies the name of the Business Warehouse Process Chain.

**Limits:** Up to 30 valid SAP characters; case-sensitive

**Example:** ZPAK\_6C7ZW2JAXS90AK71WD1CYIN

### Example: Run a Process Chain

This example runs the PC\_7A3929SH Process Chain on the SAP system.

```
insert_job: BWPCTEST
job_type: SAPBWPC
machine: SAPHTAGENT
sap_chain_id: PC_7A3929SH
```

## sap\_client Attribute—Specify SAP Client Number

The sap\_client attribute specifies the SAP client within the SAP system.

**Note:** Your agent administrator can define a default SAP client number using the jco.client.client property in the connection properties file. The sap\_client attribute overrides the default value.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP BW InfoPackage \(SAPBWIP\)](#) (see page 256)
- [SAP BW Process Chain \(SAPBWPC\)](#) (see page 258)
- [SAP Data Archiving \(SAPDA\)](#) (see page 259)
- [SAP Event Monitor \(SAPEVT\)](#) (see page 261)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP Process Monitor \(SAPPM\)](#) (see page 264)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_client: client_number`

#### *client\_number*

Specifies the client within the SAP system. The value corresponds to the General data, Target server field on the Define Background Job dialog.

**Limits:** Up to three digits

**Note:** You can specify a single digit (for example, sap\_client: 3), but JIL automatically prefixes zeroes to the value. The value will be stored as sap\_client: 003 in the database.

### Example: Specify the Numeric Client within the SAP System

This example runs the BTCTEST ABAP. The SAP client in the SAP system is 850.

```
insert_job: SAPTEST
job_type: SAPBWPC
machine: SAPHTAGENT
sap_client: 850
sap_chain_id: DEMO_CHAIN
```

## sap\_event\_id—Specify the SAP Event to Monitor or Trigger

The sap\_event\_id attribute specifies the SAP event to monitor or trigger.

### Supported Job Types

This attribute is required for the [SAP Event Monitor \(SAPEVT\) job type](#) (see page 261).

### Syntax

This attribute has the following format:

`sap_event_id: event_name`

#### *event\_name*

Specifies the name of the SAP event to monitor or trigger.

**Limits:** Up to 32 valid SAP characters; case-sensitive

### Example: Trigger an SAP event

This example triggers the SAP event named SAP\_Event.

```
insert_job: SAPMON
job_type: SAPEVT
machine: SAPAGENT
sap_event_id: SAP_Event
```

## sap\_event\_parm—Specify an SAP Event Parameter

The sap\_event\_parm attribute specifies the name of an SAP event parameter.

### Supported Job Types

This attribute is optional for the [SAP Event Monitor \(SAPEVT\) job type](#). (see page 261)

### Syntax

This attribute has the following format:

`sap_event_parm: parameters`

***parameters***

Specifies the name of the SAP event parameter.

**Limits:** Up to 64 valid SAP characters

### Example: Specify an SAP Event Parameter

This example triggers the SAP event named SAP\_Event. The event parameter is 2.

```
insert_job: SAPTRIG
job_type: SAPEVT
machine: SAPAGENT
sap_event_id: SAP_Event
sap_is_trigger: Y
sap_event_parm: 2
```

## sap\_ext\_table Attribute—Specify Data Selection Criteria

The sap\_ext\_table attribute specifies the SAP Business Warehouse InfoPackage data selection criteria.

### Support Job Type

This attribute is optional for the [SAP BW InfoPackage \(SAPBWIP\) job type](#) (see page 256).

### Syntax

This attribute has the following format:

`sap_ext_table: field_name=field, object_name=object, sign=I|E, operation=BT|EQ,  
low=low_number, high=high_number`

#### **field\_name=*field***

Specifies the Business Warehouse InfoPackage Technical Name.

**Limits:** Up to 100 characters

#### **object\_name=*object***

Specifies the Business Warehouse InfoPackage Technical InfoObject.

**Limits:** Up to 100 characters

#### **sign=I|E**

Specifies the Business Warehouse InfoPackage Technical Sign. Options are the following:

- I—Specifies include.
- E—Specifies exclude.

#### **operation=BT|EQ**

Specifies the Business Warehouse InfoPackage Technical Operation. Options are the following:

- BT—Specifies between.
- EQ—Specifies equal.

#### **low=*low\_number***

Specifies the Business Warehouse InfoPackage Technical From Value.

**Limits:** Up to 256 characters

**high=high\_number**

Specifies the Business Warehouse InfoPackage Technical To Value.

**Limits:** Up to 256 characters

**Notes:**

- To specify multiple data selection criteria, define a separate sap\_ext\_table attribute for each set of criteria. The combination of all the sap\_ext\_table attributes form a single row in the SELECTION table.
- If a keyword has no value, you can omit the keyword from the attribute.

**Example: Specify Multiple Data Selection Criteria**

This example specifies three sets of criteria in the SAP table.

```
insert_job: SAPJOB1
job_type: SAPBWIP
machine: sapagent
sap_info_pack: InfoBw2
sap_ext_table: field_name=SPTAG, object_name=0CALDAY, sign=I, operation=BT,
low=20010101,high=20060606
sap_ext_table: field_name=VBELN, object_name=0CALDAY, sign=I, operation=BT,
low=20010101,high=20060606
sap_ext_table: field_name=PKUNRE,object_name=0CALDAY, sign=I, operation=BT,
low=20010101,high=20060606
```

**Example: Specify a Row of Criteria With Select Keywords**

This example specifies one set of criteria in the SAP table. The object\_name, sign, and operation fields have no values, so they are omitted from the job definition.

```
insert_job: SAPJOB2
job_type: SAPBWIP
machine: sapagent
sap_ext_table: field_name=SPTAG,low=20010101,high=20060606
```

## sap\_fail\_msg Attribute—Specify Failure Message for SAP Job

The sap\_fail\_msg attribute specifies a string that indicates the failure of a job. If the string is found in the job's spool file, the job is considered failed.

**Supported Job Types**

This attribute is optional for the following job types:

- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_fail_msg: message`

***message***

Specifies a string that indicates the job failed. If the string is found in the job's spool file, the job is considered failed even if the job succeeds on the SAP system.

**Limits:** Up to 128 valid SAP characters; case-sensitive

### Example: Use a Failure Message to Determine a Job's Failure

This example specifies the failure message, "Internal problem". If the string is found in the SAP job log or the SAP spool output, CA Workload Automation AE treats this job as failed.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPAGENT
sap_fail_msg: Internal problem
sap_step_parms: abap_name=BTCTEST, variant=TEST
sap_job_name: SAP_JOB
```

## [\*\*sap\\_info\\_pack Attribute—Identify Name of Business Warehouse InfoPackage on SAP System\*\*](#)

The `sap_info_pack` attribute identifies the name of a Business Warehouse InfoPackage on the SAP system.

### Supported Job Type

This attribute is required for the [SAP BW InfoPackage \(SAPBWIP\) job type](#) (see page 256).

### Syntax

This attribute has the following format:

`sap_info_pack: infopackage`

***infopackage***

Specifies the name of the Business Warehouse InfoPackage.

**Limits:** Up to 30 valid SAP characters; case sensitive

#### Example: Define a Business Warehouse InfoPackage Job

This example runs the ZPAK\_6C7ZW2JAXS90AK71WD1CYIN Business Warehouse InfoPackage on the SAP system.

```
insert_job: BWIP
job_type: SAPBWIP
machine: sapagent
sap_info_pack: ZPAK_6C7ZW2JAXS90AK71WD1CYIN
```

## sap\_is\_trigger Attribute—Specify the Action to Take on an SAP Event

The sap\_is\_trigger attribute specifies whether to trigger or monitor an SAP event.

**Note:** If you do not specify the sap\_is\_trigger attribute in your job definition, the job monitors the SAP event (the default).

#### Supported Job Type

This attribute is optional for the [SAP Event Monitor \(SAPEVT\) job](#) (see page 261).

#### Syntax

This attribute has the following format:

sap\_is\_trigger: Y | N

**Y**

Triggers an SAP event.

**Note:** You must specify N in the continuous attribute.

**N**

Monitors an SAP event. This is the default.

#### Example: Trigger an SAP Event

This example triggers the SAP event named SAP\_Event. The event parameter is 2.

```
insert_job: SAPTRIG
job_type: SAPEVT
machine: SAPAGENT
sap_event_id: SAP_Event
sap_is_trigger: Y
sap_event_parm: 2
```

## sap\_job\_class Attribute—Specify SAP Job Class

The sap\_job\_class attribute specifies the SAP system job class the job will run under.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

```
sap_job_class: class
class
```

Specifies a valid SAP system job class.

**Limits:** One character

### Example: Specify an SAP System Job Class

This example runs the SAP R/3 job under the SAP system job class C.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPAGENT
sap_job_class: C
sap_job_name: SAP_JOB
sap_step_parms: abap_name=BTCTEST
```

## sap\_job\_count Attribute—Specify the Job ID of the SAP Job to Copy

The sap\_job\_count attribute specifies the ID of the job to be copied.

### Supported Job Type

This attribute is required for the [SAP Job Copy \(SAPJC\) job type](#) (see page 262).

### Syntax

This attribute has the following format:

`sap_job_count: count`

**count**

Specifies the ID of the job to be copied.

**Limits:** Up to 8 digits

### Example: Specify the ID of the Job to be Copied

This example defines an SAP Job Copy job that copies the AM job with job count 11331500 and runs the new copy.

```
insert_job: SAPJC_job
job_type: SAPJC
owner: WAAESAP@sapagent
machine: sapagent
sap_job_name: AM
sap_job_count: 11331500
```

## sap\_job\_name Attribute—Specify SAP Job Name

The sap\_job\_name attribute specifies the SAP R/3 job name.

### Supported Job Types

This attribute is required for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

This attribute is optional for the [SAP BW InfoPackage \(SAPBWIP\) job type](#) (see page 256).

### Syntax

This attribute has the following format:

`sap_job_name: job_name`

#### *job\_name*

Specifies the following:

- For SAP R/3 jobs, the SAP job name that identifies the workload in the SAP system.
- For SAP Batch Input Session jobs, the name of the ABAP job that creates the BDC session on the SAP system.
- For SAP Job Copy jobs, the name of the SAP R/3 job to be copied.
- For SAP BW InfoPackage jobs, specifies the name of the Business Warehouse InfoPackage job on the SAP system.

#### Notes:

- This attribute corresponds to the General data, Job name field on the Define Background Job dialog in SAP.
- You can use special characters, blanks, and spaces. Quotation marks must be prefixed with a backslash as an escape character.

**Limits:** Up to 32 valid SAP characters

#### **Example: Specify an SAP Job Name**

This example specifies a job name that does not contain blanks or delimiters, so the job name does not need to be enclosed in single quotation marks.

```
insert_job: SAPTEST
job_type: SAP
machine: sapagent
sap_job_name: prodrun1
sap_step_parms: abap_name=BTCTEST, step_user=J01PROD, variant=TEST
```

#### **Example: Specify an SAP Job Name That Contains Blanks and Other Characters**

This example specifies a job name that contains a space and a single quotation mark, so the job name must be enclosed in quotation marks. The quotation mark in the job name must also be duplicated.

```
insert_job: SAPTEST
job_type: SAP
machine: sapagent
sap_job_name: "Today''s Work"
sap_step_parms: abap_name=BTCTEST, step_user=J01PROD, variant=TEST
```

#### **Example: Specify a Job to be Copied**

This example copies the proctest job.

```
insert_job: SAPCOPY
job_type: SAPJC
machine: sapagent
sap_job_name: proctest
sap_job_count: 00458131
```

## sap\_lang Attribute—Specify SAP Language for Login

The sap\_lang attribute specifies the language to use to log on to the SAP system.

**Note:** Your agent administrator can define a default language using the jco.client.lang property in the connection properties file. The sap\_lang attribute overrides the default value.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP BW InfoPackage \(SAPBWIP\)](#) (see page 256)
- [SAP BW Process Chain \(SAPBWPC\)](#) (see page 258)
- [SAP Data Archiving \(SAPDA\)](#) (see page 259)
- [SAP Event Monitor \(SAPEVT\)](#) (see page 261)
- [SAP Process Monitor \(SAPPM\)](#) (see page 264)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_lang: language`

***language***

Specifies a character code representing a valid language for SAP.

**Limits:** Up to two characters; case-sensitive

**Default:** The SAP system language

**Example:** EN

**Note:** This attribute corresponds to the SAPGUI Language field on the login dialog.

### Example: Specify an SAP Language

This example defines the SAPBWPC3 job that runs the DEMO\_CHAIN1 Process Chain on the SAP system. The language used to log on to the SAP system is English.

```
insert_job: SAPBWPC3
job_type: SAPBWPC
machine: sapagent
owner: WAAESAP@sapagent
sap_chain_id: DEMO_CHAIN1
sap_lang: EN
sap_rfc_dest: SM1
```

## sap\_mon\_child Attribute—Enable Monitoring of Children Jobs

The sap\_mon\_child attribute specifies whether to monitor children jobs. Children jobs are jobs spawned by a parent job.

**Note:** If you do not specify the sap\_mon\_child attribute in your job definition, the job does not monitor children jobs (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Job Copy \(SAPCP\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

sap\_mon\_child: Y | N

**Y**

Monitors the job's children.

**N**

Does not monitor the job's children. This is the default.

### Notes:

- Your agent administrator can specify a default setting for all SAPCP and SAP jobs by setting the sap.job.children.monitor parameter in the agent's agentparm.txt file.
- The sap\_mon\_child attribute overrides the default setting specified in the agent's agentparm.txt.

#### **Example: Enable Monitoring of Children**

In this example, children spawned by SAPTEST are monitored.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPHTAGENT
sap_mon_child: Y
sap_job_name: SAP_JOB
sap_step_parms: abap_name=ZCHILDJOB1
```

## **sap\_office Attribute—Save Outgoing Documents to SAPoffice Outbox**

The sap\_office attribute specifies whether to save outgoing documents to the SAPoffice outbox of the SAP user associated with the job.

**Note:** If you do not specify the sap\_office attribute in your job definition, the job does not save outgoing documents (the default).

#### **Supported Job Types**

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP R/3 \(SAP\)](#) (see page 266)

#### **Syntax**

This attribute has the following format:

`sap_office: Y | N`

**Y**

Saves outgoing documents to the SAPoffice outbox.

**N**

Does not save outgoing documents to the SAPoffice outbox. This is the default.

#### Example: Save Outgoing Mail to the SAPoffice Outbox

In this example, outgoing mail is sent to the SAPoffice outbox of the SAP user J01Pro.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPAGENT
sap_job_name: SAP_JOB
sap_step_parms: abap_name=RSPARAM, step_user=J01PRO
sap_recipients: recipient=SU,rcp_type=SU
sap_office: Y
```

## sap\_print\_parms Attribute—Specify Print Parameters

The sap\_print\_parms attribute specifies print parameters.

### Supported Job Types

This attribute is optional for the [SAP Data Archiving \(SAPDA\) job type](#) (see page 259).

## Syntax

This attribute has the following format:

`sap_print_parms: dest=value[,keyword=value...]`

### **dest=value**

Specifies the output device name. This keyword corresponds to the SAPGUI Output device field on the Background Print Parameters dialog.

**Limits:** Up to 4 characters; case-sensitive

**Example:** dev1

### **keyword=value**

(Optional) Specifies a print parameter. You can specify multiple parameters. Separate each keyword=value pair with a comma. Options include the following:

#### **authorization=value**

Specifies the SAP authorization user for viewing the print spool list. This keyword corresponds to the SAPGUI Authorization field on the Background Print Parameters dialog.

**Limits:** Up to 12 valid SAP characters; case-sensitive

#### **banner\_page=Y | N**

Indicates whether to include an SAP cover page with the report output.

Y—Prints the cover page

N—Does not print the cover page

**Default:** N

#### **copies=value**

Specifies the number of print copies.

**Limits:** Up to 3 digits

#### **dataset=value**

Specifies the print spool data set name.

**Limits:** Up to six characters; case-sensitive

**Example:** LIST1S

#### **dept=value**

Specifies the department name to print on the cover page.

**Limits:** Up to 12 characters; case-sensitive

**expiration=value**

Specifies the number of days before the spool request is deleted.

**Limits:** 0-9

**footer=Y | N**

Indicates whether to print a footer on an SAP report.

Y—Prints the footer

N—Does not print the footer

**format=value**

Specifies an output format for an SAP report.

**Limits:** Up to 16 characters; case-sensitive

**Example:** X\_65\_80

**host\_page=Y | N | D**

Indicates whether to print an operate system page.

- Y—Prints the operating system page
- N—Does not print the operating system page
- D—Uses the SAP default

**new\_spool=Y | N**

Indicates whether to create a new spool request or append the spool request to an existing spool request with similar attributes (if any).

- Y—Creates a new spool request
- N—Appends spool request to an existing spool request

**num\_columns=value**

Specifies the line width of the list.

**Limits:** 1-255

**Example:** 251

**Note:** The maximum line width of a list for viewing on the screen is 255.

**num\_lines**

Specifies the number of lines per list page. The length of the list is determined by its content. You can use 0 or blank only when viewing the list online. You cannot use 0 or blank to format a list to be printed.

**Limits:** 0-486

**Example:** 15

**print\_imm=Y | N**

Indicates whether to print the job immediately after completion. This keyword corresponds to the SAPGUI Spool options, Print immediately field on the Background Print Parameters dialog.

- Y—Job prints immediately after completion
- N—The job output remains on spool

**Default:** N

**print\_priority**

Specifies the priority of the SAP spool request. The priority is a number between 1 (highest priority) and 9 (lowest priority).

**Default:** 5

**Example:** print\_priority=2

**prt\_arc\_mode=1 | 2 | 3**

Specifies whether the spool file is printed, archived, or both.

- Print
- Archive
- Both

**Example:** prt\_arc\_mode=Both

**Default:** Print

**recipient\_name**

Specifies the spool request recipient's name to appear on the SAP cover page. This keyword corresponds to the SAPGUI Cover sheets options, Recipient field on the Background Print Parameters dialog.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**release=Y | N**

Indicates whether the spool request associated with the ABAP is deleted after printing. This keyword corresponds to the SAPGUI Spool options, Delete after output field on the Background Print Parameters dialog.

- Y—The spool request is deleted after printing
- N—The spool request is not deleted after printing

**req\_type=value**

Specifies the print request type.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**Example:** req\_type=prtreq12

**sap\_banner=Y | N**

Indicates whether to include an SAP cover page. The SAP cover page contains information such as recipient name, department name, and format used.

- Y—Prints the SAP cover page
- N—Does not print the SAP cover page

**Example:** sap\_banner=Y

**spool\_name**

Specifies the name of the print spool.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**Example:** prtspool12

**title**

Specifies the spool request description.

**Limits:** Up to 68 characters

**Note:** The entire value can be up to 4096 characters.

**Example: Specify Print Parameters**

This example specifies the print parameters for an SAP Data Archiving job.

```
insert_job: test_SAPDA
job_class: SAP
job_type: SAPDA
machine: sapagent
owner: WAAESAP@sapserver
arc_obj_name: BC_ARCHIVE
arc_obj_variant: PRODUKTIVMODUS
sap_print_parms: dest=INHYPR502,prt_arc_mode=print,title="testing", num_lines=65,
num_columns=132, format=X_65_132, copies=1
```

## sap\_process\_client Attribute—Specify the Client to Monitor for a Specified Process

The sap\_process\_client attribute specifies the client to be monitored for the specified process.

### Supported Job Type

This attribute is optional for the [SAP Process Monitor \(SAPPM\) job type](#) (see page 264) when sap\_process\_status is set to RUNNING or STOPPED.

### Syntax

This attribute has the following format:

`sap_process_client: client`

#### ***client***

Specifies the client monitored for the specified process.

**Limits:** Up to three digits

**Example:** 800

## **sap\_process\_status Attribute—Specify the SAP Process Status to Monitor**

The `sap_process_status` attribute specifies the SAP process status to monitor.

### Supported Job Types

This attribute is required for the [SAP Process Monitor \(SAPPM\) job type](#) (see page 264).

### Syntax

This attribute has the following format:

`sap_process_status: RUNNING | STOPPED | WAITING`

#### **RUNNING**

Monitors the running status.

#### **STOPPED**

Monitors the stopped status.

#### **WAITING**

Monitors the waiting status.

### Example: Monitor for Jobs Running in the Background

This example monitors for running background jobs and checks for the SAP user named `user14`.

```
insert_job: PMTEST
job_type: SAPPM
machine: SAPHTAGENT
sap_process_status: RUNNING
sap_proc_type: BGD
sap_proc_user: user14
```

## sap\_proc\_type Attribute—Specify the SAP Process Type to Monitor

The sap\_proc\_type attribute specifies the SAP process type to monitor.

### Supported Job Type

This attribute is optional for the [SAP Process Monitor \(SAPPM\) job type](#) (see page 264).

### Syntax

This attribute has the following format:

sap\_proc\_type: BGD | DIA | ENQ | SPO | UPD | UP2

#### **BGD**

Monitors the background process type.

#### **DIA**

Monitors the dialog process type.

#### **ENQ**

Monitors the lock process type.

#### **SPO**

Monitors the spool process type.

#### **UPD**

Monitors the update process type.

#### **UP2**

Monitors the update2 process type.

### Example: Monitor for Jobs Running in the Background

This example monitors for running background jobs and checks for the SAP user named user14.

```
insert_job: PMTEST
job_type: SAPPM
machine: SAPHTAGENT
sap_process_status: RUNNING
sap_proc_type: BGD
sap_proc_user: user14
```

## sap\_proc\_user—Specify an SAP Process User

The sap\_proc\_user attribute specifies an SAP process user.

### Supported Job Type

This attribute is optional for the [SAP Process Monitor \(SAPPMM\) job type](#) (see page 264) when sap\_process\_status is set to RUNNING or STOPPED.

### Syntax

This attribute has the following format:

```
sap_proc_user: user
user
```

Specifies an SAP user name. When a process matches the specified process status, the job the process runs is checked for a match to this user.

**Limits:** Up to 80 characters; case-sensitive

### Example: Specify the User for an SAP Process

This example monitors for running background jobs and checks for the SAP user named user14.

```
insert_job: PMTEST
job_type: SAPPMM
machine: SAPHTAGENT
sap_process_status: RUNNING
sap_proc_type: BGD
sap_proc_user: user14
```

## sap\_recipients Attribute—Specify SAP Spool Recipient

The sap\_recipients attribute specifies a recipient for the SAP spool. The recipient receives the spool upon job completion or failure.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_recipients: rcp_type=type[,keyword=value...]`

#### **rcp\_type=type**

Specifies the recipient type. Options include the following:

##### **SU**

Specifies that the type is SAP user.

##### **DLI**

Specifies that the type is SAOffice distribution list.

##### **INT**

Specifies that the type is email address.

#### **keyword=value**

(Optional) Specifies a parameter for the recipient. You can specify multiple parameters. Separate each keyword=value pair with a comma. Options include the following:

##### **blind\_copy=Y | N**

Specifies whether to send the email as a blind carbon copy (bcc).

- Y—Sends the email as a blind carbon copy.
- N—Does not send the email as a blind carbon copy.

##### **copy=Y | N**

Specifies whether to send the email as a carbon copy (cc).

- Y—Sends the email as a carbon copy.
- N—Does not send the email as a carbon copy.

**express=Y | N**

Indicates whether to send instant messages to recipients if they are logged into SAP.

- Y—Sends instant messages.
- N—Does not send instant messages.

**Example:** express=Y

**Note:** This parameter only applies when rcp\_type is set to SU or DLI.

**no\_forward=Y | N**

Indicates whether the SAPoffice users can forward the document.

- Y—SAPoffice users cannot forward the document.
- N—SAPoffice users can forward the document.

**no\_print=Y | N**

Indicates whether the SAPoffice users can print the document.

- Y—SAPoffice users cannot print the document.
- N—SAPoffice users can print the document.

**recipient=value**

Specifies the target recipient's distribution address. The value corresponds to the rcp\_type value.

**Limits:** Up to 256 characters

**Examples:**

- rcp\_type=SU,recipient=SAPUSER
- rcp\_type=DLI,recipient=SAPDLIST
- rcp\_type=INT,user@example.com

**Note:** This parameter is required when rcp\_type is set to SU, DLI, or INT.

**Notes:**

- You can specify multiple spool recipients. Use one entry of sap\_recipients to define each recipient and its properties. For example, to specify two recipients user1@company.com and user2@company.com, define an sap\_recipients attribute for user1 and an sap\_recipients attribute for user2.
- The entire value can be up to 4096 characters.

### Example: Specify Multiple Spool Recipients

This example sends the SAP spool file to five recipients when the job completes or fails.

```
insert_job: sap_job
machine: sapagent
job_type: SAP
sap_job_name: BTCTEST
sap_step_parms: abap_name=BTCABAP
sap_recipients: rcp_type=INT,recipient=J01PROD@example.com
sap_recipients: rcp_type=INT,recipient=J01TEST@example.com,BlindCopy=Y
sap_recipients: rcp_type=INT,recipient=J02PROD@example.com,Copy=Y
sap_recipients: rcp_type=DLI,recipient='SAPoffice Dist List',
Express=Y,Copy=Y,NoForward=Y,NoPrint=Y
sap_recipients: rcp_type=SU,recipient='WAAESAP',BlindCopy=Y
```

## sap\_release\_option Attribute—Specify Release Option for a Job

The sap\_release\_option attribute specifies the action to take with a job after it is defined.

**Note:** If you do not specify the sap\_release\_option attribute in your job definition, the job releases the job as soon as possible (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

sap\_release\_option: I | A

I

Releases the job immediately. If no free background processing is available, the job is not released and stays in the scheduled SAP job state.

A

Releases the job as soon as possible. This is the default.

#### Example: Release a Job as Soon as Possible

This example runs the ZBDCTEST ABAP that creates a Batch Input Session (BDC ABAP) on the default SAP system defined to the agent. The job runs as soon as possible after the job is defined. After this job runs, the BDC job starts the data transfer.

```
insert_job: bdcjob
job_type: SAPBDC
machine: sapagent
owner: WAAESAP@sapagent
sap_job_name: ZBDCTEST
sap_release_option: A
sap_step_parms: abap_name="ZBDCTEST"
```

## sap\_rfc\_dest Attribute—Specify SAP Destination

The sap\_rfc\_dest attribute specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

#### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP BW InfoPackage \(SAPBWIP\)](#) (see page 256)
- [SAP BW Process Chain \(SAPBWPC\)](#) (see page 258)
- [SAP Data Archiving \(SAPDA\)](#) (see page 259)
- [SAP Event Monitor \(SAPEVT\)](#) (see page 261)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP Process Monitor \(SAPPM\)](#) (see page 264)
- [SAP R/3 \(SAP\)](#) (see page 266)

#### Syntax

This attribute has the following format:

`sap_rfc_dest: destination`

#### *destination*

Specifies the destination for the RFC connection and gateway information for an SAP R/3 system. This value corresponds to the *destination@properties* agent configuration file name containing the SAP connection information.

**Limits:** Up to 256 characters; case-sensitive

#### Example: Specify the SAP Destination

This example runs the Business Warehouse InfoPackage OPAK\_D2XZMZ1HD5WFVFL3EN1NVIT4V at the SAP destination, SM1.

```
insert_job: InfoBW2
job_type: SAPBWIP
machine: sapagent
owner: WAAESAP@sapagent
sap_job_name: InfoBW2
sap_rfc_dest: SM1
sap_info_pack: OPAK_D2XZMZ1HD5WFVFL3EN1NVIT4V
```

## sap\_step\_num Attribute—Specify Number of First Step to Copy

The sap\_step\_num attribute specifies the number of the first step to start copying job data from.

#### Supported Job Type

This attribute is optional for the [SAP Job Copy \(SAPJC\) job type](#) (see page 262).

#### Syntax

This attribute has the following format:

```
sap_step_num: step
step
```

Specifies the number of the first step to start copying job data from.

**Limits:** 0-highest step number

**Note:** If you specify 0 or 1, all steps are copied.

#### Example: Specify an SAP Step Number

This example defines an SAP Job Copy job that starts copying job data from step 1.

```
insert_job: test_SAPJC_2
job_class: SAP
job_type: SAPJC
machine: sapagent
owner: WAAESAP@sapserver
sap_job_name: AM
sap_job_count: 11331500
sap_release_option: I
sap_target_jobname: "SAPJC_DUP"
sap_step_num:1
```

## sap\_step\_parms Attribute—Specify SAP R/3 Step Specifications

The sap\_step\_parms attribute defines the SAP R/3 step specifications.

### Supported Job Types

This attribute is required for the following job types:

- [SAP R/3 \(SAP\)](#) (see page 266)
- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)

### Syntax

This attribute has the following format:

`sap_step_parms: abap_name=abap[, keyword=value...]`

#### **abap\_name=abap**

Specifies the valid SAP system ABAP name. This keyword corresponds to the SAPGUI ABAP program Name field on the Create Step dialog.

**Limits:** Up to 40 characters

**Example:** abap\_name=BTCTEST

#### **keyword=value**

(Optional) Specifies a parameter for the step. You can specify multiple parameters. Separate each keyword=value pair with a comma. Options include the following:

##### **abap\_lang=value**

Specifies a valid language for the ABAP.

**Default:** The SAP system default logon language

**Limits:** Up to 2 characters

**Examples:** abap\_lang=E

##### **arc\_client=value**

Specifies the archive link client.

**Limits:** Up to 3 digits

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**Example:** arc\_client=800

**arc\_connect=value**

Specifies the archive link communication connection.

**Limits:** Up to 14 valid SAP characters; case-sensitive

**Example:** arc\_connect=arcconnect

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_date=value**

Specifies the archive link archiving date. The format is *yyyymmdd*.

**Limits:** Up to eight digits

**Example:** arc\_date=20080630

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_doc\_class=value**

Specifies the archive link document class.

**Limits:** Up to 20 valid SAP characters; case-sensitive

**Example:** arc\_doc\_class=arcdocclass

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_doc\_type=value**

Specifies the name of the document type. This keyword corresponds to the Doc. type field on the Define Background Archive Parameters dialog.

**Limits:** Up to 10 valid SAP characters; case-sensitive

**Example:** arc\_doc\_type=archive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_format=value**

Specifies the archive output format.

**Limits:** Up to 16 valid SAP characters; case-sensitive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_host\_link=value**

Specifies the SAP archive link RPC host.

**Limits:** Up to 32 valid SAP characters; case-sensitive

**Example:** arc\_host\_link=rpchost

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_info=value***

Specifies the archive link information.

**Limits:** Up to three valid SAP characters; case-sensitive

**Example:** arc\_info=inf

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_obj\_type=value***

Specifies the type of external system archive object. This keyword corresponds to the Obj. type field on the Define Background Archive Parameters dialog.

**Limits:** Up to 10 valid SAP characters; case-sensitive

**Example:** arc\_obj\_type=archive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_path=value***

Specifies the standard archive path.

**Limits:** Up to 70 valid SAP characters; case-sensitive

**Example:** arc\_path=/export/home/archive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_printer=value***

Specifies the archive target printer.

**Limits:** Up to 4 valid SAP characters; case-sensitive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_protocol=value***

Specifies the archive storage connection protocol.

**Limits:** Up to eight valid SAP characters; case-sensitive

**Example:** arc\_protocol=prtcl12

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

***arc\_report=value***

Specifies the archive link report name.

**Limits:** Up to 30 valid SAP characters; case-sensitive

**Example:** arc\_report=Archive Report

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_service=*value***

Specifies the SAP archive link RPC service/destination.

**Limits:** Up to 32 valid SAP characters; case-sensitive

**Example:** arc\_service=rpcdest

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_storage=*value***

Specifies the archive link target storage system.

**Limits:** Up to two valid SAP characters; case-sensitive

**Example:** arc\_storage=st

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_text=*value***

Specifies the archive link Text Information field.

**Limits:** Up to 30 valid SAP characters; case-sensitive

**Example:** arc\_text='Archive text'

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_userid=*value***

Specifies the user ID for archiving the report.

**Limits:** Up to 16 valid SAP characters; case-sensitive

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**arc\_version=*value***

Specifies the archive version.

**Limits:** Up to four valid SAP characters

**Example:** arc\_version=4.1

**Note:** To use this keyword, prt\_arc\_mode must be set to Archive or Both.

**authorization=*value***

Specifies the SAP authorization user for viewing the print spool list. This keyword corresponds to the SAPGUI Authorization field on the Background Print Parameters dialog.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**banner\_page=Y | N**

Indicates whether to include an SAP cover page with the report output.

- Y—Prints the cover page
- N—Does not print the cover page

***copies=value***

Specifies the number of print copies.

**Limits:** Up to 10 digits

***dataset=value***

Specifies the print spool data set name.

**Limits:** Up to six characters; case-sensitive

**Example:** LIST1S

***dept=value***

Specifies the department name to print on the cover page.

**Limits:** Up to 12 characters; case-sensitive

***dest=value***

Specifies the output device name. This keyword corresponds to the SAPGUI Output device field on the Background Print Parameters dialog.

**Limits:** Up to 4 characters; case-sensitive

**Example:** dev1

***expiration=value***

Specifies the number of days before the spool request is deleted.

**Limits:** 0-9

***footer=Y | N***

Indicates whether to print a footer on an SAP report.

Y—Prints the footer

N—Does not print the footer

***format=value***

Specifies an output format for an SAP report.

**Limits:** Up to 16 characters; case-sensitive

**Example:** X\_65\_80

***host\_page=Y | N | D***

Indicates whether to print an operating system page.

- Y—Prints the operating system page
- N—Does not print the operating system page
- D—Uses the SAP default

**new\_spool=Y | N**

Indicates whether to create a new spool request or append the spool request to an existing spool request with similar attributes (if any).

- Y—Creates a new spool request
- N—Appends spool request to an existing spool request

**num\_columns=value**

Specifies the line width of the list.

**Limits:** 1-255

**Example:** 251

**Note:** The maximum line width of a list for viewing on the screen is 255.

**num\_lines=value**

Specifies the number of lines per list page. The length of the list is determined by its content. You can use 0 or blank only when viewing the list online. You cannot use 0 or blank to format a list to be printed.

**Limits:** 0-486

**Example:** 15

**print\_imm=Y | N**

Indicates whether to print the job immediately after completion. This keyword corresponds to the SAPGUI Spool options, Print immediately field on the Background Print Parameters dialog.

- Y—Job prints immediately after completion
- N—The job output remains on spool

**print\_priority=value**

Specifies the priority of the SAP spool request. The priority is a number between 1 (highest priority) and 9 (lowest priority).

**Default:** 5

**Example:** print\_priority=2

**prt\_arc\_mode=value**

Specifies whether the spool file is printed, archived, or both.

- Print
- Archive
- Both

**Example:** prt\_arc\_mode=Both

**recipient\_name=value**

Specifies the spool request recipient's name to appear on the SAP cover page. This keyword corresponds to the SAPGUI Cover sheets options, Recipient field on the Background Print Parameters dialog.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**Default:** Current user name

**release=Y | N**

Indicates whether the spool request associated with the ABAP is deleted after printing. This keyword corresponds to the SAPGUI Spool options, Delete after output field on the Background Print Parameters dialog.

- Y—The spool request is deleted after printing
- N—The spool request is not deleted after printing

**req\_type=value**

Specifies the print request type.

**Limits:** Up to 12 valid SAP characters; case-sensitive

**Example:** req\_type=prtreq12

**sap\_banner=Y | N**

Indicates whether to include an SAP cover page. The SAP cover page contains information such as recipient name, department name, and format used.

- Y—Prints the SAP cover page
- N—Does not print the SAP cover page

**Example:** sap\_banner=Y

**sap\_fail\_msg=value**

Specifies a string that indicates the failure of the step. If the string matches the SAP ABAP output for the step, the step is considered failed even if the step succeeds on the SAP system.

**Limits:** Up to 128 valid SAP characters; case-sensitive

**Notes:**

- This keyword does not apply to SAPBDC jobs.
- Enclose values that contain spaces in single quotation marks.

**Example:** sap\_fail\_msg='Internal problem'

**sap\_maillist="*address[,address...]*"**

Specifies one or more email addresses to send the spool list results to.

**Limits:** Up to 2048 characters; each email address can be up to 1024 characters

**Example:** sap\_maillist="user1@example.com,user2@example.com"

**sap\_success\_msg=*value***

Specifies a string that indicates the success of the step. If the string matches the SAP ABAP output for the step, the step is considered successfully completed even if the step fails on the SAP system.

**Limits:** Up to 128 valid SAP characters; case-sensitive

**Notes:**

- This keyword does not apply to SAPBDC jobs.
- Enclose values that contain spaces in single quotation marks.

**Example:** sap\_success\_msg='Program Selections'

**spool\_name=*value***

Specifies the name of the print spool.

**Limits:** Up to 16 valid SAP characters; case-sensitive

**Example:** spool\_name=prtspool12

**step\_user=*value***

Specifies the SAP R/3 system user the ABAP program runs under.

**Limits:** Up to 16 valid alphanumeric SAP characters

**Example:** step\_user=PROD01

**title=*value***

Specifies the spool request description.

**Limits:** Up to 68 characters

**variant=*value***

Specifies the variant to pass. This keyword corresponds to the SAPGUI ABAP program Variant field on the Create Step dialog.

**Limits:** Up to 14 valid alphanumeric SAP characters

**Example:** variant=test

**Notes:**

- You can specify multiple steps. Use one entry of sap\_step\_parms to define each step and its properties. For example, to specify two steps, define an sap\_step\_parms attribute for the first step and an sap\_step\_parms attribute for the second step.
- The entire value can be up to 4096 characters.

### Example: Define an SAP R/3 Job With Multiple Steps

This example runs an SAP R/3 job named SAP 3 STEPS\_@#\$. The job runs three steps. Each step runs the same ABAP but with different parameters.

```
insert_job: TEST_1
job_type: SAP
machine: localhost
job_class: c
sap_job_name: "SAP 3 STEPS_@#$"
sap_release_option: I
sap_step_parms:abap_name=BTCTEST,variant=test,arc_printer=LP01,copies=2,prt_arc_m
ode=BOTH,arc_obj_type=ARCHIVE,arc_doc_type=ARCHIVE,arc_info=inf,num_lines=65,num_
columns=80
sap_step_parms:abap_name=BTCTEST,variant=test2,arc_printer=LP01,copies=3,prt_arc_
mode=ARCHIVE,print_imm=N,release=N,sap_banner=Y,banner_page=Y,recipient_name=cybe
r,num_lines=65,num_columns=80,arc_obj_type=ARCHIVE,arc_doc_type=ARCHIVE,arc_info=
inf
sap_step_parms:abap_name=BTCTEST,variant=test,arc_printer=LP01,copies=1,prt_arc_m
ode=ARCHIVE,sap_banner=Y,banner_page=Y,recipient_name=cyber,num_lines=65,num_colu
mns=80,authorization=string2,arc_obj_type=ARCHIVE,arc_doc_type=ARCHIVE,arc_info=i
nf
```

## sap\_success\_msg Attribute—Specify Success Message for SAP Job

The sap\_success\_msg attribute specifies a string that indicates the success of the job.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_success_msg: message`

#### *message*

Specifies a string that indicates the job completed successfully. If the string is found in the job's spool file, the job is considered successfully completed even if the job fails on the SAP system.

**Limits:** Up to 128 valid SAP characters; case-sensitive.

### Example: Use a Success Message to Determine a Job's Success

This example defines the success message, "Program Selections". If the string is found in the SAP job log, CA Workload Automation AE considers the job as successful.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPHTAGENT
sap_job_name: SAP_JOB
sap_success_msg: Program Selections
sap_step_parms: abap_name=BTCTEST, step_user=J01PROD, variant=TEST
```

## sap\_target\_jobname Attribute—Specify Job Name to Copy

The sap\_target\_jobname attribute specifies the name of the target job to copy.

**Note:** If you do not specify the sap\_target\_jobname attribute in your job definition, the target job will have the same name as the source job (the default).

### Supported Job Type

This attribute is optional for the [SAP Job Copy \(SAPJC\) job type](#) (see page 262).

### Syntax

This attribute has the following format:

`sap_target_jobname: name`

***name***

Specifies the name of the target job to copy.

**Default:** The name of the source job

**Limits:** Up to 32 valid SAP characters

### Example: Specify the Name of a Job to Copy

This example defines an SAP Job Copy job that copies the job named SAPJC\_DUP.

```
insert_job: test_SAPJC_2
job_class: SAP
job_type: SAPJC
machine: sapagent
owner: WAAESAP@sapserver
sap_job_name: AM
sap_job_count: 11331500
sap_release_option: I
sap_target_jobname: "SAPJC_DUP"
group: SAPJC
```

## sap\_target\_sys Attribute—Specify SAP Application Server

The sap\_target\_sys attribute specifies the name of the SAP system application server where the SAP job is to run.

### Supported Job Types

This attribute is optional for the following job types:

- [SAP Batch Input Session \(SAPBDC\)](#) (see page 255)
- [SAP Data Archiving \(SAPDA\)](#) (see page 259)
- [SAP Job Copy \(SAPJC\)](#) (see page 262)
- [SAP Process Monitor \(SAPPM\)](#) (see page 264)
- [SAP R/3 \(SAP\)](#) (see page 266)

### Syntax

This attribute has the following format:

`sap_target_sys: application_server`

#### *application\_server*

Specifies the host computer within the SAP system architecture that is capable of running SAP system workload.

**Limits:** Up to 256 characters; case-sensitive

**Note:** If the sap\_target\_sys attribute is not defined, the SAP system will select an application server based on available resources.

### Example: Specify the SAP System Application Server

This example defines a job that runs the BTCTEST ABAP at the tst005 SAP system regardless of the entries specified in the destination properties file.

```
insert_job: SAPTEST
job_type: SAP
machine: SAPHTAGENT
sap_job_name: SAP_JOB
sap_step_parms: abap_name=BTCTEST, step_user=J01PROD, variant=TEST
sap_target_sys: tst005
```

## scp\_local\_name Attribute—Specify the Local File to Transfer

The scp\_local\_name attribute specifies a file on the agent computer to be downloaded or uploaded.

### Supported Job Type

This attribute is required for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

`scp_local_name: file_name`

#### *file\_name*

Specifies the file's destination (if downloading) or the file's source location (if uploading). The local file or files are located on the computer where the agent is installed.

**Limits:** Up to 256 characters; case-sensitive

#### Notes:

- For downloads, if the scp\_remote\_name does not contain wildcard characters, you must specify the full path and file name without wildcards. If the scp\_remote\_name does contain wildcard characters, the scp\_local\_name must specify the explicit name of the existing target directory.
- For uploads, you can use wildcards for the file name if the scp\_protocol attribute is set to SFTP. The asterisk (\*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character. If a wildcard is used in a local file name for upload, the scp\_remote\_name attribute is ignored. A wildcard transfer is equivalent to an mget transfer using an FTP client.
- You cannot rename files if wildcards are used.
- You cannot use wildcards in the path.
- You cannot use wildcards if the scp\_protocol attribute is set to SCP.
- You cannot use a semi-colon (;) to separate file names. To transfer multiple files, you can use wildcards.

**Example: Download a File from a Remote Server to the Agent Computer**

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
owner: causer@WINAGENT
```

**Example: Upload Multiple Files from the Agent Computer to a Remote Server**

This example uploads the files in the c:\temp\upload directory to the /u1/build/uploaded directory on the aixunix server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_up_mult
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: aixunix
scp_remote_dir: /u1/build/uploaded
scp_remote_name: "*"
scp_local_name: "c:\temp\upload*"
scp_protocol: SFTP
owner: causer@WINAGENT
```

## scp\_local\_user Attribute—Specify a User ID on the Agent Computer

The scp\_local\_user attribute specifies a user ID on the computer where the agent is installed. This user ID determines the access permissions on the agent computer.

### Notes:

- To use this attribute, you must specify DOWNLOAD for the scp\_transfer\_direction attribute.
- If you do not specify the scp\_local\_user attribute in your job definition, the job uses the user ID that defined the job (the default).

### Supported Job Type

This attribute is optional for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_local\_user: *user\_id*

*user\_id*

Specifies a user ID on the computer where the agent is installed.

**Default:** The user ID that defined the SCP job

**Limits:** Up to 80 characters; case-sensitive; cannot contain delimiters (such as spaces)

### Notes:

- If the scp\_transfer\_direction attribute is set to DOWNLOAD, the downloaded file is created with this user as the file owner.
- The local user ID does not require a password to be stored on the scheduling manager.
- The scp\_local\_user attribute overrides the default setting specified in the ftp.download.owner parameter in the agent's agentparm.txt.

#### Example: Download a File from a Remote Server to the Agent Computer

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
owner: causer@WINAGENT
scp_local_user: causer@WINAGENT
```

## scp\_protocol Attribute—Specify the SCP Transfer Protocol to Use

The scp\_protocol indicates whether the SCP data transfer uses Secure File Transfer Protocol (SFTP) or regular Secure Copy (SCP).

**Note:** If you do not specify the scp\_protocol attribute in your job definition, the job uses SFTP (the default).

#### Supported Job Type

This attribute is optional for the [Secure Copy \(SCP\) job type](#) (see page 268).

#### Syntax

This attribute has the following format:

scp\_protocol: SCP | SFTP

##### SCP

Transfers files using the Secure Copy Protocol (SCP).

##### SFTP

Transfers files using Secure File Transfer Protocol (SFTP). This is the default.

#### **Example: Download a File Using SCP**

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
owner: causer@WINAGENT
```

#### **Example: Upload a File Using SFTP**

This example uploads the logs.tar file to the /u/tmp directory on the hpsupport server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_upload
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: hpsupport
scp_remote_dir: /u/tmp
scp_remote_name: logs.tar
scp_local_name: "D:\temp\logs.tar"
scp_protocol: SFTP
owner: causer@WINAGENT
```

## **scp\_remote\_dir Attribute—Specify a Remote Directory**

The scp\_remote\_dir attribute specifies the file's remote source directory (if downloading) or the file's remote destination directory (if uploading).

### **Supported Job Type**

This attribute is required for the [Secure Copy \(SCP\) job type](#) (see page 268).

## Syntax

This attribute has the following format:

`scp_remote_dir: directory`

### *directory*

Specifies the file's remote source directory (if downloading) or the file's remote destination directory (if uploading).

**Limits:** Up to 256 characters; case-sensitive

## [Example: Upload Multiple Files from the Agent Computer to a Directory on a Remote Server](#)

This example uploads the files in the c:\temp\upload directory to the /u1/build/uploaded directory on the aixunix server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_up_mult
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: aixunix
scp_remote_dir: /u1/build/uploaded
scp_remote_name: "*"
scp_local_name: "c:\temp\upload*"
scp_protocol: SFTP
owner: causer@WINAGENT
```

## scp\_remote\_name Attribute—Specify a Remote File Name

The scp\_remote\_name attribute specifies the file's source location (if downloading) or the file's destination (if uploading).

### Supported Job Type

This attribute is required for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_remote\_name: *file\_name*

#### *file\_name*

Specifies the file's source location (if downloading) or the file's destination (if uploading). The remote file or files are located on the computer designated by the scp\_server\_name attribute value.

**Limits:** Up to 256 characters; case-sensitive

#### Notes:

- For downloads, you can use wildcards for the file name if the scp\_protocol attribute is set to SFTP. The asterisk (\*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character. If a wildcard is used, the scp\_local\_name (the target) must refer to a directory. A wildcard transfer is equivalent to an mget transfer using an FTP client.
- You cannot use wildcards if the scp\_protocol attribute is set to SCP.
- You cannot rename files if wildcards are used.
- You cannot use a semi-colon (;) to separate file names.

### Example: Download a File from a Remote Server to the Agent Computer

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
owner: causer@WINAGENT
```

**Example: Upload a File from the Agent Computer to a Remote Server**

This example uploads the logs.tar file to the /u/tmp directory on the hpsupport server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_upload
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: hpsupport
scp_remote_dir: /u/tmp
scp_remote_name: logs.tar
scp_local_name: "D:\temp\logs.tar"
scp_protocol: SFTP
owner: causer@WINAGENT
```

## scp\_server\_name Attribute—Specify a Remote Server Name

The scp\_server\_name attribute specifies a remote server name.

### Supported Job Type

This attribute is required for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_server\_name: *server*

*server*

Specifies the DNS name or IP address of a remote server.

**Limits:** Up to 256 characters; case-sensitive

**Example:** 172.24.36.107 (IPv4) or 0:0:0:0:FFFF:192.168.00.00 (IPv6)

### Example: Specify a Remote Server Name

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
owner: causer@WINAGENT
```

## scp\_server\_port Attribute—Specify a Remote Server Port

The scp\_server\_port specifies the port number of the remote server.

**Note:** If you do not specify the scp\_server\_port attribute in your job definition, the job uses port 22 (the default).

### Supported Job Type

This attribute is optional for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_server\_port: *port*

***port***

Specifies the port number of the remote server.

**Default:** 22

**Limits:** 0-65535

### Example: Specify the Port Number of the Remote Server

This example defines a Secure Copy Protocol job that downloads a file from a remote server using port 22.

```
insert_job: scp_download
job_type: SCP
machine: WINAGENT
scp_transfer_direction: DOWNLOAD
scp_server_name: chi-linux
scp_remote_dir: /root
scp_remote_name: install.log
scp_local_name: "C:\temp\install.log1"
scp_protocol: SCP
scp_server_port: 22
owner: causer@WINAGENT
```

## scp\_target\_os Attribute—Specify the Remote Operating System Type

The scp\_target\_os specifies the remote operating system type, which is used to determine the path separator on the remote system.

**Note:** If you do not specify the scp\_target\_os attribute in your job definition, the job uses the UNIX type (the default).

### Supported Job Type

This attribute is optional for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_target\_os: UNIX | Windows | OpenVMS | Tandem | I5OS | MVS

#### UNIX

Specifies that the remote operating system is UNIX. This is the default.

#### **Windows**

Specifies that the remote operating system is Windows.

#### **OpenVMS**

Specifies that the remote operating system is OpenVMS.

#### **Tandem**

Specifies that the remote operating system is Tandem.

#### **I5OS**

Specifies that the remote operating system is i5/OS.

#### **MVS**

Specifies that the remote operating system is MVS.

### Example: Specify the Remote Operating System

This example defines a Secure Copy job that downloads a file from a remote UNIX server.

```
insert_job: scp_download
job_type: scp
machine: localhost
owner:ftpuser@ftpserver
scp_local_name: "c:\temp\scp.out"
scp_remote_dir: /etc
scp_remote_name: profile.CA
scp_server_name: ftpserver
scp_target_os: unix
scp_transfer_direction: download
```

## scp\_transfer\_direction Attribute—Specify the File Transfer Direction

The scp\_transfer\_direction specifies the file transfer direction between the agent computer and the remote server.

**Note:** If you do not specify the scp\_transfer\_direction attribute in your job definition, the job downloads the specified file (the default).

### Supported Job Type

This attribute is optional for the [Secure Copy \(SCP\) job type](#) (see page 268).

### Syntax

This attribute has the following format:

scp\_transfer\_direction: UPLOAD | DOWNLOAD

#### UPLOAD

Transfers files from the agent computer to the remote server.

#### DOWNLOAD

Transfers files from the remote server to the agent computer. This is the default.

#### **Example: Upload Files Using SFTP**

This example uploads the files in the c:\temp\upload directory to the /u1/build/uploaded directory on the aixunix server. The job uses the Secure File Transfer Protocol (SFTP).

```
insert_job: sftp_up_mult
job_type: SCP
machine: WINAGENT
scp_transfer_direction: UPLOAD
scp_server_name: aixunix
scp_remote_dir: /u1/build/uploaded
scp_remote_name: "*"
scp_local_name: "c:\temp\upload*"
scp_protocol: SFTP
owner: causer@WINAGENT
```

## **search\_bw Attribute—Define the Time Used to Search Backwards for a z/OS Manual Job**

The search\_bw attribute defines the number of hours that the agent does a backward search for a manually-submitted job.

#### **Supported Job Type**

This attribute is optional for the [z/OS Manual \(ZOSM\) job type](#) (see page 284).

#### **Syntax**

This attribute has the following format:

```
search_btw: hours
```

***hours***

Defines the number of hours that the agent does a backward search for a manually-submitted job.

**Limits:** Up to 5 numeric digits

**Example: Search One Hour Backwards**

This example searches one hour backwards for a manually-submitted job.

```
insert_job: test_Z0SM
job_type: Z0SM
machine: zosagent
owner: user1@ca11
search_bw: 01
zos_jobname: CYBJAK11
```

## send\_notification Attribute—Specify Whether to Send a Notification When a Job Completes

The send\_notification attribute specifies whether to send a notification when the job you are defining completes.

**Note:** If you do not specify the send\_notification attribute in your job definition, the job does not send a notification when it completes (the default).

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`send_notification: y | n | f`

**y**

Sends a notification when the job completes with any status. You must also specify the notification\_id and notification\_msg attributes.

**Note:** You can specify 1 instead of y.

**n**

Does not send a notification. This is the default.

**Note:** You can specify 0 instead of n.

**f**

Sends a notification only if the job completes with a FAILURE status.

### Example: Send a Notification When the Job Completes

This example sends a notification when the job completes, regardless of its status:

```
send_notification: 1
notification_id: ccsuser1
notification_msg: The server appears to be down.
```

### Example: Send a Notification Only When the Job Completes with a FAILURE Status

This example sends a notification only when the job completes with a FAILURE status:

```
send_notification: f
```

## service\_desk Attribute—Specify Whether to Open a CA Service Desk Ticket When a Job Fails

The service\_desk attribute specifies whether to open a CA Service Desk ticket (request or incident) when the job you are defining completes with a FAILURE status. When a job is defined to open a service desk ticket, the CA Workload Automation AE scheduler initiates the opening of the ticket during terminal status processing. The scheduler prepares and sends the ticket. Messages are written to the scheduler log indicating whether the ticket was sent and processed successfully.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

service\_desk: y | n

y

Opens a CA Service Desk request when the job fails.

**Note:** You can specify 1 instead of y.

n

Does not open a CA Service Desk request when the job fails. This is the default.

**Note:** You can specify 0 instead of n.

**Note:** Only the service\_desk attribute is required to open a service desk ticket for a job. You can also specify the following optional service desk-related attributes in your job definition:

- svcdesk\_desc
- svcdesk\_pri
- svcdesk\_imp
- svcdesk\_sev
- svcdesk\_attr

If the optional attributes are not set, the job uses the CA Workload Automation AE service desk template values set in CA Service Desk. This template is included in the CA Service Desk installation. Before initiating a service desk ticket, you must ensure that the web services for CA Service Desk are active.

#### **Example: Initiate a Service Desk Incident**

This example initiates a service desk incident with a priority of 1 for a job named service\_desk\_on\_failure\_1.

```
insert_job: service_desk_on_failure_1
machine: localhost
command: false
owner: user@localhost
service_desk: y
svcdesk_pri: 1
svcdesk_desc: "service_desk_on_failure_1 has failed."
```

#### **Example: Initiate a Service Desk Request**

This example initiates a service desk request with an impact of 3 and a severity of 4 for a job named service\_desk\_on\_failure\_2.

```
insert_job: service_desk_on_failure_2
machine: localhost
command: false
owner: user@localhost
service_desk: y
svcdesk_imp: 3
svcdesk_sev: 4
svcdesk_desc: "service_desk_on_failure_2 has failed."
```

## **service\_name Attribute—Specify the Web Service Name Within the Target Namespace**

The service\_name attribute specifies the web service name within the target namespace in a Web Service job.

#### **Supported Job Type**

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

#### **Syntax**

This attribute has the following format:

```
service_name: service
service
```

Specifies the web service name within the target namespace.

**Limits:** Up to 255 characters; case-sensitive

**Note:** In a Web Service job, if you specify the WSDL\_URL attribute but not the endpoint\_URL attribute, you must specify both the service\_name and port\_name attributes. For the job to run successfully without the endpoint\_URL attribute, the agent must be running on the same computer as the application server such as WebLogic or JBoss. If you specify both the WSDL\_URL and endpoint\_URL attributes, then the service\_name and port\_name attributes are optional.

#### Example: Invoke Web Service to Get Stock Quote

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webservicex.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a java.lang.String object, which maps to the XML type string in the return namespace <http://www.webservicex.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

## shell Attribute—Specify a UNIX Shell to Run a Script or Command

The shell attribute specifies the shell used to run a script or command in a CMD job on a UNIX computer.

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204) on UNIX.

### Syntax

This attribute has the following format:

`shell: shell_name`

#### *shell\_name*

Specifies the name of the shell used to execute the script or command file. You can specify any of the following UNIX shells:

- `/bin/ksh` (Korn shell)
- `/bin/sh` (Bourne shell)
- `/bin/bash` (Bourne again shell)
- `/bin/csh` (C shell)
- `/usr/local/bin/perl` (Perl shell)

**Limits:** Up to 20 characters

### Notes:

- The shell attribute applies to CMD jobs on UNIX only. This attribute is ignored for CMD jobs on Windows.
- The shell can be specified in different places. To run a UNIX script, the agent uses, in the following order, the shell specified in:
  1. The shell attribute (if specified in the job definition)
  2. The first line of the script (if the shell attribute is not specified)
  3. The `oscomponent.defaultshell` parameter in the `agentparm.txt` file (if not specified in the shell attribute or in the script)
  4. The user default shell defined in the user profile (if not specified in one of the previous three locations)
- The shell attribute only applies to the command attribute. The script is invoked using the specified shell.

- The shell attribute does not apply to the profile attribute. The job profile is always sourced using the job owner's default shell, which is set for the user in the etc/passwd file.
- If the oscomponent.checkvalidshell parameter in the agent's agentparm.txt file is set to true (the default), all shells that are used must be specified using the oscomponent.validshell parameter. If the shell you want to use in your job definition or script is not defined on the agent, the job fails. For more information about specifying valid shells in the agentparm.txt file, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- Different shells provide different facilities and have different characteristics. Some functions are specific to certain shells and may be incompatible with other shells.

#### Example: Specify a Shell to Run a Script

This example uses the C shell to run the sort script.

```
insert_job: MFGSORT
job_type: CMD
machine: unixagt
command: /mfg/test/sort
shell: /bin/csh
```

## sp\_arg Attribute—Specify a Parameter to Pass to a Stored Procedure

The sp\_arg attribute specifies a parameter to pass to a database stored procedure.

### Supported Job Type

This attribute is optional for the [Database Stored Procedure \(DBPROC\) job type](#) (see page 209).

### Syntax

This attribute has the following format:

`sp_arg: ignore=yes|no, name=arg_name, argtype=IN|OUT|INOUT, datatype=type,  
value=arg_value`

#### **ignore=yes | no**

Specifies whether the value is returned. Options are the following:

- yes—Specifies that the value is not returned.
- no—Specifies that the value is returned.

**Note:** This argument only applies to OUT parameters.

#### **name=arg\_name**

Specifies the name of the parameter.

#### **argtype=IN | OUT | INOUT**

Specifies the parameter type. Options are the following:

- IN—Specifies that the parameter is an input parameter.
- OUT—Specifies that the parameter is an output parameter.
- INOUT—Specifies that the parameter is an input-output parameter.

#### **datatype=type**

Specifies the [database data type of the parameter](#) (see page 672).

The following data types are supported:

- CHAR
- VARCHAR
- LONGVARCHAR
- NUMERIC
- DECIMAL
- BIT

- BOOLEAN
- TINYINT
- SMALLINT
- INTEGER
- BIGINT
- REAL
- FLOAT
- DOUBLE
- DATE
- TIME
- TIMESTAMP

**value=arg\_value**

Specifies the value of the parameter.

**Limits:** Up to 256 characters; case-sensitive

**Notes:**

- This value applies to input parameters only.
- Enclose values that contain spaces in quotation marks.

**Notes:**

- The entire value can be up to 4096 characters.
- To specify multiple parameters to pass to a stored procedure, define a separate sp\_arg attribute for each parameter. Specify the parameters in the order that they need be passed to the stored procedure.

**Example: Pass Two Parameters to a Stored Procedure**

This example passes two parameters to the UPDATE\_INV stored procedure. QUANTITY is an input parameter of type integer with a value of 3000, and NEW\_INV is an output parameter of type integer.

```
insert_job: SPJOB
job_type: DBPROC
machine: DB_agent
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
sp_name: UPDATE_INV
sp_arg: ignore=yes, name=QUANTITY, argtype=IN, datatype=INTEGER, value=3000
sp_arg: ignore=yes, name=NEW_INV, argtype=OUT, datatype=INTEGER
```

## Supported Data Types

The following table lists the supported data types for different databases:

**Note:** The data types listed in the database columns are defined in the procedure definition within the database. In the job definition, use the corresponding JDBC data type from the first column.

| JDBC Data Type | Valid Input Format               | Oracle                    | Microsoft SQL Server      | DB2             |
|----------------|----------------------------------|---------------------------|---------------------------|-----------------|
| CHAR           | Plain text                       | CHAR                      | CHAR                      | CHAR            |
| VARCHAR        | Plain text                       | VARCHAR2(x)<br>VARCHAR(x) | VARCHAR(x)                | VARCHAR(x)      |
| LONGVARCHAR    |                                  | Not supported             | Not supported             | Not supported   |
| NUMERIC        | Number                           | NUMBER,<br>NUMERIC        | NUMERIC                   | NUMERIC         |
| DECIMAL        | Number                           | DECIMAL                   | DECIMAL                   |                 |
| BIT            |                                  | Not supported             | BIT                       | Not supported   |
| BOOLEAN        |                                  | Not supported             | Not supported             | Not supported   |
| TINYINT        |                                  | Not supported             | TINYINT                   | Not supported   |
| SMALLINT       | Number                           | SMALLINT                  | SMALLINT                  | SMALLINT        |
| INTEGER        | Number                           | INTEGER                   | INTEGER<br>INT            | INTEGER         |
| BIGINT         | Number                           | Not supported             | BIGINT                    | BIGINT          |
| REAL           | Number<br>[dot Number]           | REAL                      | REAL                      | REAL            |
| FLOAT          | Number<br>[dot Number]           | FLOAT                     | FLOAT                     | FLOAT           |
| DOUBLE         |                                  |                           | FLOAT                     | DOUBLE, DECIMAL |
| DATE           | yyyy-mm-dd                       | DATE                      | Not supported             | DATE            |
| TIME           | hh:mm:ss                         |                           | Not supported             | TIME            |
| TIMESTAMP      | yyyy-mm-dd<br>hh:mm:ss.fffffffff | TIMESTAMP                 | datetime<br>smalldatetime | TIMESTAMP       |

## Handling Unsupported Data Types

Although JDBC provides functionality for Booleans, the Oracle database driver does not. To handle unsupported data types such as Booleans, you can call a wrapper procedure in the job definition. The wrapper procedure converts the inputted values to values supported by the database driver and calls the desired stored procedure from inside the database.

### Example: Run a Wrapper Procedure Named `boolwrap`

In the following example, the agent runs a wrapper procedure named `boolwrap`, which converts the inputted integer value to a Boolean and calls the `boolproc` stored procedure.

```
CREATE OR REPLACE PROCEDURE boolproc(x boolean) AS BEGIN
[...]
END;

CREATE OR REPLACE PROCEDURE boolwrap(x int) AS BEGIN
IF (x=1) THEN
 boolproc(TRUE);
ELSE
 boolproc(FALSE);
END IF;
END;
```

## sp\_name Attribute—Specify Stored Procedure or Stored Function to Run

The `sp_name` attribute specifies the database stored procedure to run. If you use an Oracle or SQL Server database, you can also use the `sp_name` attribute to specify a stored function.

### Supported Job Type

This attribute is required for the [Database Stored Procedure \(DBPROC\) job type](#) (see page 209).

### Syntax

This attribute has the following format:

`sp_name: stored_procedure`

#### ***stored\_procedure***

Specifies the stored procedure to run. You can specify a stored function if you use an Oracle or SQL Server database.

**Limits:** Up to 300 characters; case-sensitive

**Note:** The agent does not support stored functions (also called user-defined functions) on DB2.

#### **Example: Run a Stored Procedure**

This example invokes the PAYROLL procedure and checks whether the employee name (ename) starts with John. The procedure is stored in a SQL Server database named ORDERS. The database user ID is set to the user who invokes jil to define the job (the default owner). When the procedure runs, the parameter ename, which is both an input and output parameter of type VARCHAR, is passed with the value John Evans.

```
insert_job: SPJOB
job_type: DBPROC
machine: DB_agent
connect_string: "jdbc:sqlserver://myhost:1433;DatabaseName=ORDERS"
sp_name: PAYROLL
success_criteria: ename=John.*
sp_arg: ignore=no, name=ename, argtype=INOUT, datatype=VARCHAR, value=John Evans
```

## **sql\_command Attribute—Specify an SQL Statement to Run**

The sql\_command attribute specifies an SQL statement to run against a database table.

#### **Supported Job Type**

This attribute is required for the [SQL job type](#) (see page 272).

#### **Syntax**

This attribute has the following format:

```
sql_command: sql_statement
```

***sql\_statement***

Specifies the SQL statement to run against a database table.

**Limits:** Up to 4078 characters; case-sensitive

**Example:** select emp\_name from emp where emp\_name like 'S%%'

**Note:** The value can contain any SQL statement including SELECT, DELETE, UPDATE, or INSERT.

**Note:** To use the sql\_command attribute, you must know how to construct an SQL query.

**Example: Return Data from a Table that Match a Condition**

This example queries a database table named emp for an emp\_name value that starts with S, followed by at least two characters. If the result starts with S, followed by one other character, the job completes. Otherwise, the job fails. The job's output is written to c:\temp\sql\_job1\_out.log.

```
insert_job: sql_job1
job_type: SQL
machine: SQL_server1
sql_command: select emp_name from emp where emp_name like 'S%'
connect_string:"jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
success_criteria: emp_name=S.*
destination_file: "c:\temp\sql_job1_out.log"
```

## start\_mins Attribute—Define the Minutes to Run a Job

The start\_mins attribute defines the number of minutes after each hour at which the job should start on the days specified in the days\_of\_week attribute or on the dates specified in the calendar identified in the run\_calendar attribute.

**Note:** Times set in the start\_times and start\_mins attributes override times set in a custom calendar. The date\_conditions attribute controls the usage of this attribute.

You cannot specify both the start\_times and start\_mins attributes in the same job definition. If you do, CA Workload Automation AE ignores both attributes and generates an error.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`start_mins: mins [, mins]...`

#### *mins*

Defines the number of minutes after each hour to run the job.

**Limits:** This value can be a number in the range 0 to 59. This value can contain up to 255 characters, and you can use multiple lines without specifying a continuation character. Separate multiple values with commas.

**Default:** No start time is set.

### Example: Specify a Job to Run at a Quarter Past and Half Past Each Hour

This example specifies that the job should run at a quarter past and half past each hour:

```
date_conditions: y
start_mins: 15, 30
```

## start\_times Attribute—Define the Time of the Day to Run a Job

The start\_times attribute defines the times of day, in 24-hour format, at which the job should start on the days specified in the days\_of\_week attribute or on the dates specified in the calendar identified in the run\_calendar attribute.

**Note:** Times set in the start\_times and start\_mins attributes override times set in a custom calendar. The date\_conditions attribute controls the usage of this attribute.

You cannot specify both the start\_times and start\_mins attributes in the same job definition. If you do, CA Workload Automation AE ignores both attributes and generates an error.

When CA Workload Automation AE writes a job definition containing the start\_times attribute to the database within one minute of the specified start time, the job definition takes effect at the specified start time on the following day. However, when the product writes a job definition containing the start\_times attribute to the database two or more minutes before the specified start time, the job definition takes effect at the specified start time on the same day.

For example, if you define a job to run daily at 12:00 and you edit its definition at 11:59, the job does not run with your changes until the following day at 12:00. However, if you edit the job's definition at 11:58 or before, the job runs with your changes at 12:00 on the same day.

### Supported Job Types

This attribute is optional for all job types.

## Syntax

This attribute has the following format:

```
start_times: "time [, time]..."
```

### time

Defines the times at which the job should run, in "*hh:mm*" format, where *hh* specifies hours (in 24-hour format), *mm* specifies minutes, and multiple times are separated by commas. You must enclose the specified times in quotation marks ("") or an error will result. CA Workload Automation AE generates an error if you define the days\_of\_week or run\_calendar attributes for the job, but you do not specify values in the start\_times attribute.

**Limits:** This value can contain up to 255 characters. You can use multiple lines without specifying a continuation character.

**Default:** No start time is set.

**Note:** Because JIL parses on the combination of keyword followed by a colon, you must use an escape character (backslash) with any colons used in an attribute statement's value. For example, to define the start\_time value, enter:

```
start_times: 10\:00, 14\:00
```

**Example: Specify the Time of the Day to Run a Job**

This example specifies that the job should run at 10:00 a.m. and 2:00 p.m. on every specified day or date:

```
date_conditions: y
start_times: "10:00, 14:00"
```

or

```
date_conditions: y
start_times: 10\:00, 14\:00
```

## std\_err\_file Attribute—Redirect the Standard Error File

The std\_err\_file attribute identifies the location to redirect the standard error file for the job you are defining. The error output can be redirected to a file or a blob.

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

### Syntax

This attribute has the following format:

`std_err_file: file_name | blob_name`

#### *file\_name*

Defines the path and file name where you want to redirect all standard error output.

If you are running jobs across platforms, the scheduler of the issuing instance controls the default behavior. For UNIX, the default is to append this file. For Windows, the default is to overwrite this file. To overwrite the file, enter `>` as the first character in the value. To append the file, enter `>>` as the first characters in the value.

**Limits:** Up to 255 characters; it cannot contain spaces between the `>` or `>>` characters and the full path or file name

#### Notes:

- The job owner must have write permission to the specified file on the client computer.
- When specifying standard input, output, and error file locations, we recommend that you enter absolute paths to the files instead of relative paths or file names alone.
- You can use variables exported from the job profile or from global variables in the path name specification. Enclose variables referenced in the job profile in braces (for example, “ `${PATH}` ”). Use the format  `$$ {global_name}`  for global variables.

#### UNIX:

- This value overrides the instance-wide setting for the `AutoInstWideAppend` parameter in the configuration file.
- This value overrides the machine-specific setting for the `AutoMachWideAppend` parameter in the `/etc/auto.profile` file.

**Windows:**

- This value overrides the instance-wide setting for the Append stdout/stderr field in the CA Workload Automation AE Administrator utility.
- When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.
- If the Windows command specified in the job definition does not exist, the job does not run. The standard error file is not created and the job log indicates that the error file is not found.

**CA WA Agent Default:** The agent directory  
(*installation\_directory/SystemAgent/agent\_name/*)

**Legacy Agent Default:** /dev/null (UNIX) or NULL (Windows)

***blob\_name***

Specifies the format, type, and name associated with the blob. Blobs may be associated with the job or created as globally accessible. They can be created in either binary or text formats. Unlike file output, blobs may not be appended to.

Blob names and types are identified by the following keywords:

**\$\$blob**

Captures the output and stores it as a binary formatted error blob associated with the current job.

**\$\$blobt**

Captures the output and stores it as a textual formatted error blob associated with the current job.

**\$\$glob.<global\_blob\_name>**

Captures the output and stores it as a binary formatted blob under the *global\_blob\_name* specified.

**\$\$globt.<global\_blob\_name>**

Captures the output and stores it as a textual formatted blob under the *global\_blob\_name* specified.

**Example: Specify the File to Receive Standard Error File Output for a Job on UNIX**

This example redirects the job's standard error file output to the /tmp/test.err file.

```
insert_job: unix_err
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_err_file: /tmp/test.err
```

### Example: Append New Information to the Standard Error File on UNIX

This example appends new information to the standard error file.

```
insert_job: unix_appenderr
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_err_file: >>/tmp/test.err
```

### Example: Specify a File Name with a Global Variable on UNIX

This example redirects the job's standard error file output to a file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: unix_glob_var
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_err_file: /tmp/$${Today}.err
```

### Example: Create a Unique Identifier on UNIX

This example redirects the job's standard error file to a file with a unique name. The \$\$ characters append the process ID to the file name.

```
insert_job: unix_unique_name
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_err_file: /tmp/my_file.$$
```

To embed the process ID in the middle of the file name, you must enter a dot, slash, or space following the \$\$ characters so that CA Workload Automation AE does not interpret the string following the \$\$ as a global variable. The following example embeds the process ID before the .err characters:

```
std_err_file: /tmp/my_file.$$.err
```

You can use braces ( {} ) to separate the \$\$ characters from the string err, as follows:

```
std_err_file: /tmp/my_file.$${}err
```

Otherwise, CA Workload Automation AE attempts to interpret err as a global variable. If it cannot find a global variable called err, CA Workload Automation AE drops that part of the file name, creating a file named my\_file (because \$\$err is null).

### **Example: Specify the Standard Error File for a Job on Windows**

This example redirects the standard error file to the C:\tmp\TEST.ERR file.

```
insert_job: win_err
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_err_file: "C:\tmp\TEST.ERR"
```

### **Example: Append New Information to the Standard Error File on Windows**

This example appends new information to the standard error file.

```
insert_job: win_appenderr
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_err_file: ">>C:\tmp\TEST.ERR"
```

### **Example: Specify a File Name with a Global Variable on Windows**

This example redirects the job's standard error file output to a file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: win_globalvar
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_err_file: "C:\tmp\$$Today.err"
```

### **Example: Create a Unique Identifier on Windows**

This example redirects the job's standard error file to a file with a unique name. The %AUTOPID% characters append the process ID to the file name.

```
insert_job: win_uniquename
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_err_file: "C:\tmp\my_file.%AUTOPID%"
```

### Example: Specify a Standard Error File with an Incorrect Windows Command

Suppose that the bdproc command does not exist on the winagent computer. This example does not run and the standard error file is not created. The job log indicates that the error file is not found.

```
insert_job: win_err
job_type: CMD
machine: winagent
command: bdproc
std_err_file: "C:\tmp\error_file.txt"
```

### Example: Store Standard Error File Output as a Job Blob in Textual Format

This example captures the standard error output and stores it as a textual format blob associated with the job.

```
insert_job: report_job1
job_type: CMD
machine: localhost
command: myapplication
std_err_file: $$blobt
```

### Example: Store Standard Error File Output as a Global Blob in Textual Format

This example captures the standard error output and stores it as a textual format global blob named “some\_stderr\_data”.

```
insert_job: report_job2
job_type: CMD
machine: localhost
command: myapplication
std_err_file: $$globt.some_stderr_data
```

## std\_in\_file Attribute—Redirect the Standard Input File

The std\_in\_file attribute specifies where you want to redirect the standard input file from. The standard input file can be a file or a blob.

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

## Syntax

This attribute has the following format:

`std_in_file: file_name | blob_name`

### *file\_name*

Specifies the path and file name where you want to redirect standard input from.

**Limits:** Up to 255 characters

#### Notes:

- The job owner must have read permission to the specified file on the client computer.
- When specifying standard input, output, and error file locations, we recommend that you enter absolute paths to the files instead of relative paths or file names alone.
- You can use variables exported from the job profile or from global variables in the path name specification. Enclose variables referenced in the job profile in braces (for example, “\${PATH}”). Use the format \${global\_name} for global variables.

#### Windows:

- If you do not specify a full path name, the path defaults to the job owner's home directory.
- When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.

**Default:** Do not redirect standard input.

### *blob\_name*

Specifies the format, type, and name associated with the blob. Blobs may be associated with the job or created as globally accessible. They may be created in either binary or text formats.

Blob names and types are identified by the following keywords:

#### `$$blobt`

Uses the highest version of textual formatted input job blob associated with the current job. These blobs are manually added through jil, using the `insert_blob` subcommand.

#### `$$blob.<jobname>`

Uses an existing binary formatted output blob associated with the specified job.

**\$\$blobt.<jobname>**

Uses an existing textual formatted output blob associated with the specified job.

**\$\$glob.<global\_blob\_name>**

Uses an existing binary formatted blob under the *global\_blob\_name* specified.

**\$\$globt.<global\_blob\_name>**

Uses an existing textual formatted blob under the *global\_blob\_name* specified.

**Note:**

- You cannot use **\$\$blob** keyword to specify the use of the current job's input blob.
- You can define a job to use the output blob of its previous run as input. To accomplish this, you must define the job so that the name of the job is in the **std\_in\_file** attribute, using either **\$\$blob.<job name>** or **\$\$blobt.<job name>** keywords. After you define the job, apply a one-time override of the **std\_in\_file** attribute, so that the job reads from a local file on the computer on its first run.

**Example: Specify the Standard Input File on UNIX**

This example specifies the /tmp/test.in file as the standard input file.

```
insert_job: unix_stdin
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_in_file: /tmp/test.in
```

**Example: Specify a File Name that Contains a Global Variable on UNIX**

This example specifies a standard input file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: unix_globalvar
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_in_file: /tmp/$${Today}.in
```

### **Example: Specify a Standard Input File on Windows**

This example specifies the C:\tmp\TEST.IN file as the standard input file.

```
insert_job: win_stdin
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_in_file: "C:\tmp\TEST.IN"
```

### **Example: Specify a File Name that Contains a Global Variable on Windows**

This example specifies a standard input file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: win_globalvar
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_in_file: "C:\tmp\$$Today.in"
```

### **Example: Retrieve Standard Input File from a Job Blob in Textual Format**

This example defines a CMD job that creates and uses an input job blob. When the job is defined, a blob is created using the textual data contained in the my\_args.txt file. The blob is associated with the job. When the job runs, it uses the created blob as input and treats the blob data as textual data.

```
insert_job: report_job1
job_type: CMD
command: myapplication
machine: localhost
std_in_file: $$blobt
insert_blob: report_job1
blob_file: my_args.txt
```

### Example: Retrieve Standard Input File from Another Job in Textual Format

This example defines two Command jobs. The report\_job1 job creates an output job blob in textual format. This output job blob is associated with the job. The report\_post\_job1 job retrieves the output job blob created from report\_job1 and uses it as input.

```
insert_job: report_job1
job_type: CMD
command: myapplication
std_out_file: $$blobt
machine: localhost

insert_job: report_post_job1
job_type: CMD
command: my_other_application
machine: WinProd1
std_in_file: $$blobt.report_job1
```

## std\_out\_file Attribute—Redirect the Standard Output File

The std\_out\_file attribute specifies where you want to redirect the standard output file. The standard output can be redirected to a file or a blob.

### Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204).

### Syntax

This attribute has the following format:

```
std_out_file: file_name | blob_name
file_name
```

Defines the path and file name where you want to redirect all standard output.

If you are running jobs across platforms, the scheduler of the issuing instance controls the default behavior. For UNIX, the default is to append this file. For Windows, the default is to overwrite this file. To overwrite the file, enter `>` as the first character in the value. To append the file, enter `>>` as the first characters in the value.

**Limits:** Up to 255 characters; it cannot contain spaces between the `>` or `>>` characters and the full path or file name

**Notes:**

- The job owner must have write permission to the specified file on the client computer.
- When specifying standard input, output, and error file locations, we recommend that you enter absolute paths to the files instead of relative paths or file names alone.
- You can use variables exported from the job profile or from global variables in the path name specification. Enclose variables referenced in the job profile in braces (for example, “\${PATH}”). Use the format \$\$*global\_name* for global variables.

**UNIX:**

- This value overrides the instance-wide setting for the AutoInstWideAppend parameter in the configuration file.
- This value overrides the machine-specific setting for the AutoMachWideAppend parameter in the /etc/auto.profile file.

**Windows:**

- This value overrides the instance-wide setting for the Append stdout/stderr field in the CA Workload Automation AE Administrator utility.
- When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.

**Default:** /dev/null (UNIX) or NULL (Windows)

***blob\_name***

Specifies the format, type, and name associated with the blob. Blobs may be associated with the job or created as globally accessible. They can be created in either binary or text formats. Unlike file output, blobs may not be appended to.

Blob names and types are identified by the following keywords:

**\$\$blob**

Captures the output and stores it as a binary formatted output blob associated with the current job.

**\$\$blobt**

Captures the output and stores it as a textual formatted output blob associated with the current job.

**\$\$glob.<*global\_blob\_name*>**

Captures the output and stores it as a binary formatted blob under the *global\_blob\_name* specified.

**\$\$globt.<global\_blob\_name>**

Captures the output and stores it as a textual formatted blob under the *global\_blob\_name* specified.

**Example: Specify the File to Receive Standard Output for a Job on UNIX**

This example specifies the /tmp/test.out file to receive standard output for the job.

```
insert_job: unix_stdout
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_out_file: /tmp/test.out
```

**Example: Append New Information to the Output File on UNIX**

This example appends new information to the /tmp/test.out output file.

```
insert_job: unix_appendout
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_out_file: >>/tmp/test.out
```

**Example: Specify a File Name with a Global Variable on UNIX**

This example redirects the job's standard output to a file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: unix_globalvar
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_out_file: /tmp/$${Today}.out
```

#### **Example: Create a Unique Identifier on UNIX**

This example redirects the job's standard output to a file with a unique name. The \$\$ characters append the process ID to the file name.

```
insert_job: unix_unique
job_type: CMD
machine: unixagent
command: /usr/common/backup
std_out_file: /tmp/my_file.$$
```

To embed the process ID in the middle of the file name, you must enter a dot, slash, or space following the \$\$ characters so that CA Workload Automation AE does not interpret the string following the \$\$ as a global variable. The following example embeds the process ID before the .mary characters:

```
std_out_file: /tmp/my_file.$$.mary
```

You can use braces ( {} ) to separate the \$\$ characters from the string mary, as follows:

```
std_out_file: /tmp/my_file.$${}mary
```

Otherwise, CA Workload Automation AE attempts to interpret mary as a global variable. If it cannot find a global variable called mary, CA Workload Automation AE drops that part of the file name, creating a file named my\_file (because \$\$mary is null).

#### **Example: Specify the File to Receive Standard Output for a Job on Windows**

This example specifies the C:\tmp\TEST.OUT file to receive standard output for the job.

```
insert_job: win_stdout
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_out_file: "C:\tmp\TEST.OUT"
```

#### **Example: Append New Information to the Output File on Windows**

This example appends new information to the C:\tmp\TEST.OUT output file.

```
insert_job: win_appendout
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_out_file: ">>C:\tmp\TEST.OUT"
```

**Example: Specify a File Name with a Global Variable on Windows**

This example redirects the job's standard error file output to a file whose path contains the global variable named Today. You can use the sendevent command to set the value of a global variable to today's date.

```
insert_job: win_globalvar
job_type: CMD
machine: winagent
command: "C:\COMMON\Backup"
std_out_file: "C:\tmp\${Today}.out"
```

**Example: Store Standard Output File Output as a Job Blob in Textual Format**

This example captures the standard output and stores it as a textual format blob associated with the job.

```
insert_job: report_job1
job_type: CMD
machine: localhost
command: myapplication
std_out_file: $$blobt
```

**Example: Store Standard Output File Output as a Global Blob in Textual Format**

This example captures the standard output and stores it as a textual format global blob named “some\_stdout\_data”.

```
insert_job: report_job1
job_type: CMD
machine: localhost
command: myapplication
std_out_file: $$globt.some_stdout_data
```

## success\_codes Attribute—Define Exit Codes to Indicate Job Success

The success\_codes attribute defines which exit codes indicate job success. For example, you can define the success\_codes attribute to be 20-30. If a job returns an exit code between 20 and 30, the job is considered to have completed successfully.

**Note:** If you do not specify the success\_codes attribute in your job definition, the scheduling manager interprets an exit code of zero (0) as job success (the default). Any exit code other than zero is interpreted as job failure.

### Supported Job Types

This attribute is optional for the following job types:

- [Command \(CMD\)](#) (see page 204)
- [i5/OS \(I5\)](#) (see page 224)

### Syntax

This attribute has the following format:

`success_codes: exitcodes`

#### *exitcodes*

Defines which exit codes indicate job success. Specify a single exit code (for example, 4), a range of exit codes (for example, 0-9999), or a list of multiple exit codes or ranges separated by commas.

**Example:** 1,150,-20--10,20-30

**Note:** If you specify multiple exit codes, enter the most specific codes first followed by the ranges.

**Limits:** Up to 256 characters

**Default:** 0

### Example: Define Multiple success\_codes

Suppose that you want a job named CMDJOB to run the procjob.exe file. The job is considered to have completed successfully if the exit code is in the 20-30 or 100-150 ranges.

```
insert_job: CMDJOB
job_type: CMD
machine: winagt
command: "c:\temp\procjob.exe"
success_codes: 20-30,100-150
```

## success\_criteria Attribute—Specify the Evaluation Criteria for a Return String

The success\_criteria attribute defines a regular expression that is used to evaluate a string returned by an SQL statement or a stored procedure.

### Supported Job Types

This attribute is optional for the following job types:

- [Database Stored Procedure \(DBPROC\)](#) (see page 209)
- [SQL](#) (see page 272)

### Syntax

This attribute has the following format:

```
success_criteria: regexp
regexp
```

Defines a regular expression that is used to evaluate a return string. If the return string matches the regular expression, the job completes successfully. Otherwise, the job fails.

**Note:** Each return string includes the field name from the SELECT statement and its value, separated by an equal sign (=). For example, consider the query SELECT ORD\_NUM FROM SALES. To match order number A2976, specify the regular expression ORD\_NUM=A2976. Specifying the regular expression A2976 does not match any return string causing the job to fail. You can also specify the regular expression .\*A2976, which matches any return string that ends with A2976.

### Notes:

- The success\_criteria attribute only applies to SQL queries that are SELECT statements.
- To compose a regular expression, follow the rules for Java class java.util.regex.Pattern. You can find these rules using a Google search for java pattern.

- You can use escape sequences in the regular expression. The following characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\).
  - \—backward slash
  - [—opening bracket
  - \*—asterisk
  - |—vertical bar
  - {—opening brace
  - +—plus sign
  - (—opening parenthesis
  - ^—caret (circumflex)
  - ?—question mark
  - )—closing parenthesis
  - \$—dollar sign
  - .—period

For example, to match the characters \*.\* literally, specify \\*\.\\* in your regular expression. The backward slashes escape the characters' special meanings.

#### **Example: Define Basic Success Criteria**

This example queries the Inv\_List table. If the SQL query returns a PartNo that begins with IDG, the job completes successfully.

```
insert_job: QRY1
job_type: SQL
machine: DB_agent
sql_command: SELECT PartNo FROM Inv_List WHERE Stock > 30
success_criteria: PartNo=IDG.*
connect_string: "jdbc:sqlserver://myhost:1433;DatabaseName=ORDERS"
destination_file: "c:\log\qry1.txt"
```

### Example: Define a More Complex Success Criteria

This example queries the Proc\_List table. If the SQL query returns an ID that matches the specified regular expression, the job completes successfully.

```
insert_job: QRY2
job_type: SQL
machine: DB_agent
sql_command: SELECT ID FROM Proc_List WHERE Orders > 30
success_criteria: ID=\w{2,4}
connect_string: "jdbc:db2://172.31.255.255:50000/SAMPLE"
destination_file: /export/home/user1/log/qry2.txt
```

The regular expression is interpreted as follows:

- \w—A word character [a-zA-Z0-9]
- {2,4}—Match at least 2 times, but not more than 4 times

To illustrate the last item (2, 4), consider the syntax:

```
ID=b1{1,3}c
```

Evaluating this expression yields the following conditions:

- The line contains the text b1.
- Numeric 1 should exist at least once, but not more than three times.
- The specified string must be followed by the letter c.

## success\_pattern Attribute—Specify a Success Pattern Using Regular Expression Logic

The success\_pattern attribute specifies a regular expression that is used to check for an expected value in a JMX and Web Service job. If the output matches the success pattern, the job completes; otherwise, it fails.

### Supported Job Types

This attribute is optional for the following job types:

- [JMX-MBean Attribute Get \(JMXMLG\)](#) (see page 232)
- [JMX-MBean Attribute Set \(JMXMLAS\)](#) (see page 233)
- [JMX-MBean Operation \(JMXMLOP\)](#) (see page 237)
- [Web Service \(WBSVC\)](#) (see page 276)

### Syntax

This attribute has the following format:

`success_pattern: pattern`

#### *pattern*

Specifies a regular expression to use as a filter. To match any character zero or more times, add `.*` to the regular expression. To match an entire line of text using part of the line, add `.*` before and after the part. For example, you can use `.*web.*` to match "this is a web service job".

**Limits:** Up to 256 characters; case-sensitive

**Example:** <StockQuotes><Stock><Symbol>CA.\*

**Notes:** To compose a regular expression, follow the rules for Java class `java.util.regex.Pattern`. You can find these rules on the Internet by searching for "java pattern".

**Example: Validate an Email Address in a Web Service Job**

Suppose that you want to invoke a web service that validates an email address. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.net/ValidateEmail.asmx?wsdl>. The job calls the IsValidEmail operation within the ValidateEmail web service. When the job invokes the web service, the email address is passed to the operation. If the email address is valid, the operation returns true and the job completes successfully. If the email address is invalid, the operation returns false and the job fails.

```
insert_job: subscribe
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: ValidateEmail
port_name: ValidateEmailSoap
wsdl_operation: IsValidEmail
WSDL_URL: "http://www.webservicex.net/ValidateEmail.asmx?wsdl"
endpoint_URL: "http://www.webservicex.net/ValidateEmail.asmx"
web_parameter: xsd\string="john.smith@example.com"
return_class_name: java.lang.Boolean
return_xml_name: boolean
return_namespace: "http://www.webservicex.net"
success_pattern: true
```

## svcdesk\_attr Attribute—Define CA Service Desk Request Attributes

The svcdesk\_attr attribute defines CA Service Desk request attributes and values to set in the CA Service Desk request generated when you have set the service\_desk attribute to **y** or **1** and the job you are defining completes with a FAILURE status.

**Note:** For information about valid attributes and values, see the *CA Service Desk Modification Guide*.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`svcdesk_attr: attribute/value[,attribute/value,attribute/value...]`

#### **attribute**

Identifies a CA Service Desk attribute to override with the specified value in the generated request.

#### **value**

Defines the value to which to set the associated CA Service Desk request attribute.

**Limits:** The svcdesk\_attr value can be up to 255 alphanumeric characters in length. You must use a slash to separate *attribute* from *value*, and you must separate each *attribute/value* pair with a comma.

#### **Example: Set the CA Service Desk Priority Attribute**

This example sets the CA Service Desk priority attribute to 3, as an alternative to using the svcdesk\_pri attribute:

```
svcdesk_attr: priority/3
```

#### **Example: Set Four Attributes at a Time**

This example sets four attributes at once, such as priority=4, severity=1, impact=2, and description=This is a test:

```
svcdesk_attr: priority/4,severity/1,impact/2,description/"This is a test"
```

#### **Example: Set the CA Service Desk Urgency Attribute**

This example sets the CA Service Desk urgency attribute equal to 1:

```
svcdesk_attr: urgency/urg:1100
```

Only users with advanced knowledge of Service Desk should attempt to use an attribute, such as urgency with svcdesk\_attr, as it requires specific predefined values, such as urg:1100. (urg:1104 indicates an urgency of 5).

**Note:** For more information, see the Service Desk documentation.

## svcdesk\_desc Attribute—Define the Message to Include in the CA Service Desk Request

The svcdesk\_desc attribute defines the message to include in the CA Service Desk request generated when you have set the service\_desk attribute to **y** or **1** and the job you are defining completes with a FAILURE status.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

svcdesk\_desc: *value*

#### **value**

Specifies a description to include in the CA Service Desk request generated when the job fails.

**Limits:** Up to 255 alphanumeric characters

### Example: Define a Message to Include in the CA Service Desk Request

This example defines the description of the problem that caused a job to fail:

svcdesk\_desc: The server appears to be down.

## svcdesk\_imp Attribute—Specify the Impact Level of the CA Service Desk Request

The svcdesk\_imp attribute specifies the impact level to assign the CA Service Desk request generated when you have set the service\_desk attribute to **y** or **1** and the job you are defining completes with a FAILURE status. The impact level indicates how much you expect the request to affect work being performed.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

svcdesk\_imp: *level*

#### *level*

Specifies the impact level of the request.

**Limits:** This value is one of the following:

**0**

None. The request has no impact.

**1**

High

**2**

Medium-High

**3**

Medium

**4**

Medium-Low

**5**

Low

### Example: Set the Impact Level of the CA Service Desk Request

This example sets the impact level of a request generated when the job fails to medium:

```
svcdesk_imp: 3
```

## svcdesk\_pri Attribute—Specify the Priority Level of the CA Service Desk Request

The svcdesk\_pri attribute specifies the priority level of the CA Service Desk request generated when you set the service\_desk attribute to **y** or **1**, and the job you are defining completes with a FAILURE status.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`svcdesk_pri: level`

#### *level*

Specifies the priority level of the request.

**Limits:** Set this value to one of the following:

**0**

Indicates an unassigned priority.

**1**

Indicates that the priority level is high.

**2**

Indicates that the priority level is medium high.

**3**

Indicates that the priority level is medium.

**4**

Indicates that the priority level is medium low.

**5**

Indicates that the priority level is low.

### Example: Set the Priority Level of the CA Service Desk Request

This example sets the priority level of a request generated when the job fails to medium:

```
svcdesk_pri: 3
```

## svcdesk\_sev Attribute—Specify the Severity Level of the CA Service Desk Request

The svcdesk\_sev attribute specifies the severity level to assign the CA Service Desk request generated when you have set the service\_desk attribute to **y** or **1** and the job you are defining completes with a FAILURE status. The severity level indicates how much you expect the request to affect other users.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

svcdesk\_sev: *level*

#### *level*

Specifies the severity level of the request.

**Limits:** This value is one of the following:

**0**

Low. This is the same as setting the value to **5**.

**1**

High

**2**

Medium-High

**3**

Medium

**4**

Medium-Low

**5**

Low

### Example: Set the Severity Level of the CA Service Desk Request

This example sets the severity of a request generated when the job fails to medium:

```
svcdesk_sev: 3
```

## tablename Attribute—Specify the Database Table to Monitor

The tablename attribute specifies the name of the database table to monitor for the changes specified in the Database Monitor or Database Trigger job.

### Supported Job Type

This attribute is required for the following job types:

- [Database Monitor \(DBMON\)](#) (see page 207)
- [Database Trigger \(DBTRIG\)](#) (see page 211)

### Syntax

This attribute has the following format:

tablename: *table*

#### ***table***

Specifies the name of the database table to monitor for changes.

**Limits:** Up to 100 characters; case-sensitive

### Example: Monitor for a Decrease in the Number of Rows in a Table

This example monitors the Inventory\_List table for a decrease in the number of rows. If the number of rows decreases, the job completes. The database user ID is set to the user who invokes jil to define the job (the default owner).

```
insert_job: dbmon_job
job_type: DBMON
machine: DB_agent
tablename: Inventory_List
monitor_type: DECREASE
connect_string: "jdbc:db2://172.31.255.255:50000/SAMPLE"
```

## target\_namespace Attribute—Specify a Target Namespace

The target\_namespace attribute specifies the target namespace in a Web Service job.

### Supported Job Type

This attribute is required for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

target\_namespace: *target\_namespace*

#### ***target\_namespace***

Specifies the target namespace used for the names of messages, port type, binding, and services defined in the WSDL for the web service. Complex data types such as arrays require the target namespace.

**Limits:** Up to 256 characters; case-sensitive

### Example: Specify Target Namespace for Getting Stock Quote

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webservicex.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a java.lang.String object, which maps to the XML type string in the return namespace <http://www.webservicex.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

## term\_run\_time Attribute—Specify the Maximum Runtime

The term\_run\_time attribute specifies the maximum run time (in minutes) that the job you are defining should require to finish normally. If the job runs longer than the specified time, CA Workload Automation AE terminates it.

**Windows Note:** Windows does not support the concept of process groups. When you issue a KILLJOB event for a job that runs an executable (\*.exe), KILLJOB kills the process specified in the command definition. When you issue a KILLJOB event for a job that runs something other than an \*.exe (for example, \*.bat, \*.cmd, or \*.com), KILLJOB terminates only the CMD.EXE process that CA Workload Automation AE used to launch the job. The Job Status is set according to the return code of the killed CMD.EXE process and can be one of the following: SUCCESS, FAILURE, or TERMINATED. Processes launched by user applications or batch (\*.bat) files are not killed.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`term_run_time: mins`

#### *mins*

Defines the maximum number of minutes the job should ever require to finish normally.

**Limits:** 0-527,040

**Default:** 0 (the job is allowed to run forever)

### Example: Terminate the Job if it Runs Beyond the Maximum Run Time

This example specifies that the product should terminate the job if it runs for more than two hours:

`term_run_time: 120`

## **text\_file\_filter Attribute—Specify a Text String to Search For**

The `text_file_filter` attribute specifies the text to search for.

### **Supported Job Type**

This attribute is required for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### **Syntax**

This attribute has the following format:

`text_file_filter: textstring`

#### ***textstring***

Defines the text string to search for. You can specify the text string as a regular expression.

**Limits:** Up to 1024 characters; case-sensitive

### **Notes:**

- The OMTF job searches the entire file specified by the `text_file_name` attribute. When the job finds the first match to the text string, the job completes.
- If `monitor_mode: CONTINUOUS` is specified in the job definition, and the job finds one or more occurrences of the searched text, the first trigger takes place at the first occurrence only. Subsequently, the job only searches for new lines that are added to the text file. In other words, the job resumes the search from the first line following the last line of the file from the previous search.
- To compose a regular expression, follow the rules for Java class `java.util.regex.Pattern`. You can find these rules using a Google search for `java pattern`.

- You can use escape sequences in the regular expression. The following characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\).
  - \—back slash
  - |—vertical bar
  - (—opening parenthesis
  - )—closing parenthesis
  - [—opening bracket
  - {—opening brace
  - ^—caret (circumflex)
  - \$—dollar sign
  - \*—asterisk
  - +—plus sign
  - ?—question mark
  - .—period

For example, to match the characters \*.\* literally, specify \\*\.\.\* in your regular expression. The backward slashes escape the characters' special meanings.

#### **Example: Monitor a File for a String**

This example shows a search of the text file DATA SOURCE NAME for the text string EVENT ""COMPUTER()''. If this string is found, the job completes.

```
insert_job: textfile_job1
job_type: OMTF
machine: monagt
text_file_name: DATA SOURCE NAME
text_file_filter: "EVENT ""COMPUTER()''
text_file_mode: LINE
lower_boundary: 123
upper_boundary: 876
```

### Example: Monitor a File for a String Using Special Characters

In this example, because the text string contains escape sequences, back slashes must precede the special characters asterisk (\*), period (.), and zed (Z). The required text string is a wild card search for the text string "=jars\*.\*" that must appear at the end of the line (\Z).

```
insert_job: textfile_job2
job_type: OMTF
machine: monagt
text_file_name: /export/home/agentdir/agentparm.txt
text_file_filter: "=jars*\.*\Z"
text_file_mode: LINE
lower_boundary: 1
upper_boundary: 110
monitor_mode: now
```

### Example: Monitor a File for a String using a Regular Expression

In this example, the text string contains regular expression pattern matching syntax. The search range is also a regular expression as indicated by the text\_file\_mode attribute.

```
insert_job: textfile_job3
job_type: OMTF
machine: monagt
text_file_name: /export/home/agentdir/agentparm.txt
text_file_filter: ^\w{4,10}\.
text_file_mode: REGEX
lower_boundary: \A\W\sE
```

The regular expression can be interpreted as follows:

- ^ or \A — match only at the beginning of string (line)
- \Z or \$ — match only at the end of string
- \w — a word character [a-zA-Z0-9]
- \W — a non-word character
- \s — a whitespace character
- {4,10} — match at least 4 times but not more than 10 times

To illustrate the last item (4, 10), consider the syntax:

```
text_file_filter: b1{1,3}c
```

Evaluating this expression yields the following conditions:

- The line contains the text b1.
- Numeric 1 should exist at least once, but not more than three times.
- The specified text string must be followed by the letter c.

#### **Example: Monitor a Windows Text File for a Windows Path**

Because back slashes are interpreted as an escape sequence, the two back slashes in the text string, C:\\Program Files, must each be preceded by a back slash, giving four back slashes in the text\_file attribute below.

```
insert_job: textfile_job4
job_type: OMTF
machine: monagt
text_file_name: /export/home/agentdir/agentparm.txt
text_file_filter: "C:\\\\Program Files"
text_file_mode: LINE
lower_boundary: 1
upper_boundary: 200
monitor_mode: now
```

## **text\_file\_filter\_exists Attribute—Specify Whether to Monitor for the Existence of Text**

The text\_file\_filter\_exists attribute specifies whether the job monitors for the existence of a text string in a text file.

**Note:** If you do not specify the text\_file\_filter\_exists attribute in your job definition, the job checks whether the text string exists (the default).

#### **Supported Job Type**

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

#### **Syntax**

This attribute has the following format:

text\_file\_filter\_exists: TRUE | FALSE

#### **TRUE**

Monitors whether a text string exists in a specified text file. If the text string exists, the job completes successfully or an alert is triggered (if monitoring continuously). This is the default.

#### **FALSE**

Monitors whether a text string does not exist in a specified text file. If the text string does not exist, the job completes successfully. If the text string exists, the job fails.

**Note:** To use the FALSE setting, you must specify NOW in the monitor\_mode attribute.

### Example: Check Whether a String Does Not Exist in a File

This example searches lines 1 to 200 of the transmitter.log file for the text string "Warning". If the string is *not* found, the job completes successfully. If the string is found, the job fails.

```
insert_job: textfile_job
job_type: OMTF
machine: monagt
text_file_name: /export/home/log/transmitter.log
text_file_filter: Warning
text_file_mode: LINE
lower_boundary: 1
upper_boundary: 200
text_file_filter_exists: FALSE
monitor_mode: NOW
```

## text\_file\_mode Attribute—Specify the Search Mode

The text\_file\_mode attribute specifies the search mode when monitoring a text file.

This attribute is used with the lower\_boundary and upper\_boundary attributes, which define the search boundaries in the text file.

**Note:** If you do not specify the text\_file\_mode attribute in your job definition, the job searches for the text in the specified line boundaries (the default).

### Supported Job Type

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### Syntax

This attribute has the following format:

text\_file\_mode: LINE | REGEX | DATETIME

#### LINE

Searches for a text string between line numbers that define the upper and lower boundaries to search in the text file. This is the default. The values for the lower\_boundary and upper\_boundary attributes must be numeric.

#### REGEX

Searches for a text string between regular expressions that define the lower and upper boundaries to search in the text file. The values for the lower\_boundary and upper\_boundary attributes must be regular expressions.

## DATETIME

Searches for a text string between a date range specified by the upper and lower boundaries. You must specify the time\_format attribute when using this value.

### Notes:

- You must specify the lower\_boundary attribute, upper\_boundary attribute, or both. Most searches require you to specify both attributes.
- For all search modes, the upper and lower boundaries are determined as follows:
  - If you only specify the lower\_boundary attribute, the agent searches from the lower boundary to the last line of the text file.
  - If you only specify the upper\_boundary attribute, the agent starts searching from the first line of the text file to the upper boundary.
- For regular expressions, the upper and lower boundaries are determined as follows:
  - If you specify the upper\_boundary attribute as a regular expression and there are no strings in the file that match it, the agent searches for the text string to the last line of the file.
  - If you specify the lower\_boundary attribute as a regular expression and there are no strings in the file that match it, the agent does not search for the text string.

### Example: Define the Search Mode as LINE

This example searches the c:\ca\log file in line mode. The job starts searching the content from line 143 of the file. The upper boundary is not defined, so the job searches to the last line of the file. The job completes successfully if the ERROR MESSAGE string is found.

```
insert_job: omtf_line
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 143
monitor_mode: NOW
```

**Example: Define Search Mode as REGEX**

This example searches the c:\ca\log file in regular expression mode. The lower boundary is not defined, so the job searches the content from the first line of the file to the upper boundary (a line that contains the word service). The job completes successfully if the ARCHIVE string is found.

```
insert_job: omtf_regex
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ARCHIVE
text_file_mode: REGEX
upper_boundary: service
monitor_mode: NOW
```

**Example: Define the Search Mode as DATETIME**

This example searches the /export/home/logs/transmitter.log file in date and time mode. The job searches the content between May 20, 2010 at midnight and May 27, 2010 at 11:59 p.m. The date and time values are defined using the format specified in the time\_format attribute. The job completes successfully if the transmitted string is found.

```
insert_job: omtf_timedate
job_type: OMTF
machine: monagt
text_file_name: /export/home/logs/transmitter.log
text_file_filter: transmitted
text_file_mode: DATETIME
lower_boundary: "Thu May 20 00:00:00.000 EDT 2010"
upper_boundary: "Thu May 27 23:59:59.999 EDT 2010"
time_format: "EEE MMM dd HH:mm:ss.SSS zzz yyyy"
time_position: 12
monitor_mode: NOW
```

#### **Example: Monitor a UNIX Text File for a String at the Beginning of a Line**

This example searches the /export/home/systemagent/agentparm.txt file. The search starts at the first line that contains the word "agent" at the beginning of the line (as specified by \A in the regular expression specified for lower\_boundary) and until it finds the string "level=2" at the end of a line (as specified by \Z in the regular expression specified for upper\_boundary).

```
insert_job: omtf_unix_line
job_type: OMTF
machine: monagt
text_file_name: /export/home/systemagent/agentparm.txt
text_file_filter: \.0/MAIN$
text_file_mode: REGEX
lower_boundary: \Aagent
upper_boundary: level=2\Z
monitor_mode: NOW
```

#### **Example: Monitor a UNIX Text File for a String Between Regular Expressions**

This example searches for the string "# Runner plugin parameters", which is evaluated as a regular expression. The job searches from the first line of the file to the line that contains the word service.

```
insert_job: omtf_unix_regex
job_type: OMTF
machine: monagt
text_file_name: /export/home/systemagent/agentparm.txt
text_file_filter: \A\W\sRunner
text_file_mode: REGEX
upper_boundary: service
monitor_mode: now
```

The criteria \A, \W, and the word \sRunner are evaluated as follows:

- \A means match only at the beginning of the line.
- \W means match a non-word character.
- \sRunner means look for a white space before the word Runner.

**Example: Monitor an i5/OS Text File for a String in the First 20 Lines of the File**

The following job searches for a text string "Create file failed". This job searches the DATA member for this text string between lines 1 (the lower boundary) and 20 (the upper boundary). The job completes successfully when the text string is found.

```
insert_job: omtf_i5os_line
job_type: OMTF
machine: i5agt
text_file_name: /QSYS.LIB/LIBRARY.LIB/RESULTS.FILE/DATA.MBR
text_file_filter: Create file failed
text_file_mode: LINE
lower_boundary: 1
upper_boundary: 20
monitor_mode: now
```

## text\_file\_name Attribute—Specify a Text File Name and Location

The text\_file\_name attribute specifies the name and location (path) of the text file to search.

### Supported Job Type

This attribute is required for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### UNIX and Windows Syntax

This attribute has the following format:

```
text_file_name: file_name
file_name
```

Specifies the path to and name of the text file to search.

**Limits:** Up to 256 characters; case-sensitive

**Windows Note:** When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.

**Example:** "C:\\Program Files\\agent\\prodagt"

#### **Example: Specify a Text File Name on Windows**

This example searches for a text string in the log text file on a Windows computer. The path to the text file is enclosed in quotation marks because the path contains a colon. The job searches for the string "ERROR MESSAGE" between lines 1234 and 1876. The job completes successfully if the string is found.

```
insert_job: textfile_job1
job_type: OMTF
machine: monagt
text_file_name: "c:\program files\agent\log"
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 1234
upper_boundary: 1876
monitor_mode: NOW
```

#### **Example: Specify a Text File Name on UNIX**

This example searches the /export/home/logs/transmitter.log file in date and time mode. The job searches the content between May 20, 2010 at midnight and May 27, 2010 at 11:59 p.m. The date and time values are defined using the format specified in the time\_format attribute. The job completes successfully if the transmitted string is found.

```
insert_job: omtf_timedate
job_type: OMTF
machine: monagt
text_file_name: /export/home/logs/transmitter.log
text_file_filter: transmitted
text_file_mode: DATETIME
lower_boundary: "Thu May 20 00:00:00.000 EDT 2010"
upper_boundary: "Thu May 27 23:59:59.999 EDT 2010"
time_format: "EEE MMM dd HH:mm:ss.SSS zzz yyyy"
time_position: 12
monitor_mode: NOW
```

## time\_format Attribute—Define a Time Format

The time\_format attribute defines the date and time pattern to use when searching a text file with a time stamp.

### Supported Job Type

If DATETIME is specified for the text\_file\_mode attribute, this attribute is required for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### Syntax

This attribute has the following format:

`time_format: time_format`

#### *time\_format*

Defines the date and time pattern to use when the upper and lower boundaries are specified as date and time. The upper and lower boundaries are used to search inside a log file.

**Limits:** Up to 256 characters; case-sensitive

### Notes:

- To specify the time format, construct a time pattern string that is used as a mask for searching the particular time stamp you want to find. In this pattern, all ASCII letters are reserved as pattern letters.
- The following table displays the symbols you can use to build a time format pattern:

---

### Time Format Pattern Characters

---

| Character | Definition              | Type            | Example  |
|-----------|-------------------------|-----------------|----------|
| G         | Era designator          | Text            | AD       |
| y         | Year                    | Number          | 2010     |
| M         | Month in year           | Text and Number | July; 07 |
| d         | Day in month            | Number          | 10       |
| h         | Hour in am/pm (1 to 12) | Number          | 12       |
| H         | Hour in day (0 to 23)   | Number          | 0        |
| m         | Minute in hour          | Number          | 30       |
| s         | Second in minute        | Number          | 55       |
| S         | Millisecond             | Number          | 978      |

---

| <b>Time Format Pattern Characters</b> |                         |                 |                            |
|---------------------------------------|-------------------------|-----------------|----------------------------|
| <b>Character</b>                      | <b>Definition</b>       | <b>Type</b>     | <b>Example</b>             |
| E                                     | Day in week             | Text            | Tuesday; Tues              |
| D                                     | Day in year             | Number          | 189                        |
| F                                     | Day of week in month    | Number          | 2 (2nd Wednesday in July)  |
| w                                     | Week in year            | Number          | 27                         |
| W                                     | Week in month           | Number          | 2                          |
| a                                     | AM/PM marker            | Text            | PM                         |
| k                                     | Hour in day (1 to 24)   | Number          | 24                         |
| K                                     | Hour in AM/PM (0 to 11) | Number          | 0                          |
| z                                     | Time zone               | Text            | EST; Eastern Standard Time |
| Z                                     | RFC 822 time zone       | Sign and Number | -0500                      |
| '                                     | Escape for test         | Delimiter       |                            |
| "                                     | Single quote            | Literal         | '                          |

- The following table displays sample time format patterns for July 10, 2010 at 12:08 p.m. Eastern Standard Time:

| <b>Format Pattern</b>          | <b>Result</b>                 |
|--------------------------------|-------------------------------|
| "yyyy.MM.dd G 'at' hh:mm:ss z" | 2010.07.10 AD at 12:08:56 EST |
| "EEE, MMM d, "yy"              | Wed, July 10, '10             |
| "h:mm a"                       | 12:08 PM                      |
| "hh 'o'clock' a, zzzz"         | 12 o'clock PM, EST            |
| "K:mm a, z"                    | 12:08 PM, EST                 |
| "yyyy.MMMMMdd GGG hh:mm aaa"   | 2010.July.10 AD 12:08 PM      |

### Example: Define a Time Format

In this example, the time\_format attribute specifies the time pattern format that is applied to the values specified by the lower\_boundary and upper\_boundary attributes. The time values are enclosed in quotation marks because they contain colons.

The values for the time\_format attribute are as follows:

- MM—month in the year
- dd—day in the month
- yyyy—year
- HH:mm:ss.SSS—hours, minutes, seconds, and milliseconds.
- zZ—time zone and RFC 822 time zone

The job completes successfully if the string is found.

```
insert_job: timeformat_job
job_type: OMTF
machine: monagt
text_file_name: /export/home/agentdir/log/transmitter.log
text_file_filter: UNAGR
text_file_mode: DATETIME
time_format: "MM/dd/yyyy HH:mm:ss.SSS zZ"
lower_boundary: "08/17/2010 00:00:00.000 EST-0500"
upper_boundary: "08/25/2010 23:59:59.999 EST-0500"
monitor_mode: now
```

## time\_position Attribute—Specify the First Column of a Time Stamp

The time\_position attribute defines the first column of the time stamp in a text file.

**Note:** To use this attribute, you must specify DATETIME for the text\_file\_mode attribute.

### Supported Job Type

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

### Syntax

This attribute has the following format:

`time_position: timepos`

***timepos***

Specifies the first column of the time stamp in the log file.

**Limits:** Up to 5 digits

**Examples:** 12, 12345

### Example: Define the First Column of a Time Stamp

In this example, the time\_format attribute provides the time pattern format that is applied to the values specified by the lower\_boundary and upper\_boundary attributes. The first column of the time stamp is 12. The time values are enclosed in quotation marks because they contain colons.

The values for the time\_format attribute are as follows:

- h:mm—hour in a.m./p.m., minute in hour
- a—a.m./p.m. marker

The job completes successfully if the string is found.

```
insert_job: timeformat_job
job_type: OMTF
machine: monagt
monitor_mode: now
text_file_name: /export/home/logs/transaction.log
text_file_filter: UNAGR
text_file_mode: DATETIME
time_format: "h:mm a"
time_position: 12
lower_boundary: "10:08 PM"
upper_boundary: "12:30 PM"
monitor_mode: now
```

## timezone Attribute—Define the Time Zone

The `timezone` attribute defines the time zone for the job you are defining. For example, if you define a start time of 01:00 for a job running on a machine in Denver, and set `timezone` to San Francisco (which is in the Pacific time zone, one hour earlier than Denver), the job starts at 2:00 a.m. Denver time.

The start event for jobs with time-based starting conditions that do not specify a time zone is scheduled based on the time zone under which the scheduler is running.

### Supported Job Types

This attribute is optional for all job types.

### Syntax

This attribute has the following format:

`timezone: zone`

#### **zone**

Specifies the time zone for the job. The job's time settings are based on this time zone, regardless of the server location. Specify a string that corresponds to an entry in the `uko_timezones` table. On UNIX, you can specify any time zone recognized by the operating system.

**Limits:** Up to 50 characters; valid characters are a-z, A-Z, decimal digits, slash (/), hyphen (-), and underscore (\_); not case-sensitive; do not include spaces or periods

**Note:** Enclose values that contain colons in quotation marks. If you do not enclose a value that contains a colon in quotation marks, the colon is interpreted as a delimiter.

**Examples:** "IST-5:30", MountainTime (instead of Mountain Time), Solomons (instead of Solomon Is)

### Notes:

- Some regions may change to daylight saving time (DST) on different days and times than the time zone that the server is running in. Jobs scheduled to start in a DST time zone will run one hour earlier (or later) until the time zone in the job definition and the time zone where the server is running are both in the same time scheme (either DST or Standard Time).
- To display all the entries in the `uko_timezones` table, enter the `autotimezone -l` command at the instance command prompt.

**UNIX Notes:**

- The ujo\_timezones table contains entries for all the common time zones maintained by the operating system and for many cities in the United States.
- The scheduling manager interprets the string and matches it to a time zone value on your operating system. If the string is not found, the ujo\_timezones table is read up to five times to resolve a zone value. If the zone value is not resolved after five attempts, the job fails. We recommend that you use a time zone value available from your operating system to help ensure that the scheduling manager is using the same values as other applications running on that computer.

**Windows Notes:**

- The ujo\_timezones table contains entries for all the cities and time zones supported by Windows and for many cities in the world.
- Windows time zones are located in the Date/Time applet in the Control Panel. Most of the values in the Time Zone menu of the Date/Time applet are cities. You can specify any of those values as the time zone in your job definition. For example, if you want a job to run in the “Bogota, Lima, Quito,” time zone as shown on Windows, you can specify “Bogota,” “Lima,” or “Quito.”

**Example: Set the Time Zone to Chicago Time**

This example sets the time zone for a job definition to Chicago time:

```
timezone: Chicago
```

**Example: Set the Time Zone to Pacific Time**

This example sets the time zone for a job definition to Pacific time:

```
timezone: US/Pacific
```

## trigger\_cond Attribute—Specify a Condition to Monitor For

The trigger\_cond attribute specifies a condition to monitor the database for. The job completes when the condition is met.

### Supported Job Type

This attribute is optional for the [Database Trigger \(DBTRIG\) job type](#) (see page 211).

### Syntax

This attribute has the following format:

`trigger_cond: condition`

#### *condition*

Specifies the condition to monitor in the database.

- For Oracle and DB2, this condition is the WHEN clause.
- For SQL Server, this condition is the IF clause.

**Limits:** Up to 256 characters; case-sensitive

**Note:** For the specific database syntax, refer to your database vendor's documentation.

### Example: Specify a Trigger Condition for Oracle

This example monitors the Inventory\_List table for updates. The job connects to the Oracle database named ORDERS. When the INVENTORY\_LIST is updated, if the number of units of productA has fallen below 1000, the job completes.

```
insert_job: dbtrig_oracle
job_type: DBTRIG
machine: dbagent
tablename: Inventory_List
trigger_type: UPDATE
trigger_cond: new.ProductA < 1000
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

**Example: Specify a Trigger Condition for SQL Server**

This example monitors the sales table for changes to the ord\_date and qty columns. The job runs under the sa user, who owns the table and is authorized to create triggers on the database or schema the table belongs to. The job completes only when both columns have changed.

```
insert_job: dbtrig_sqlsvr
job_type: DBTRIG
machine: dbagent
dbtype: MSSQL
owner: sa@myhost
connect_string: "jdbc:sqlserver://myhost:1433;DatabaseName=pubs"
tablename: sales
trigger_type: UPDATE
trigger_cond: UPDATE(ord_date) and UPDATE(qty)
```

**Example: Monitor a SQL Server Database Table for Added Rows with a Trigger Condition**

This example monitors the sales table for added rows. The job runs under the sa user, who owns the table and is authorized to create triggers on the database or schema the table belongs to. The job connects to the default database resource location defined on the agent. When the qty for inserted title ID TC7777 is greater than or equal to 20, the job completes.

```
insert_job: dbtrig3
job_type: DBTRIG
machine: DB_agent
dbtype: MSSQL
trigger_type: INSERT
trigger_cond: (select QTY from INSERTED where TITLE_ID='TC7777')>=20
tablename: sales
owner: sa@myhost
```

#### **Example: Specify a Trigger Condition for IBM DB2**

This example monitors the STAFF table for changes. The job connects to the SAMPLE database and runs under the entadm user, who is authorized to create triggers on the database or schema the table belongs to. The job completes when the table has changed.

```
insert_job: dbtrig_db2
job_type: DBTRIG
machine: dbagent
dbtype: DB2
owner: entadm@myhost
connect_string:"jdbc:db2://172.31.255.255:50000/SAMPLE"
tablename: STAFF
dbtype: DB2
trigger_type: UPDATE
```

## **trigger\_type Attribute—Specify the Type of Database Change to Monitor For**

The trigger\_type attribute specifies the type of database change to monitor for.

**Note:** If you do not specify the trigger\_type attribute in your job definition, the job monitors for the insertion of a row in the database table (the default).

#### **Supported Job Type**

This attribute is optional for the [Database Trigger \(DBTRIG\) job type](#) (see page 211).

#### **Syntax**

This attribute has the following formats:

```
trigger_type: INSERT
trigger_type: DELETE
trigger_type: UPDATE
trigger_type: INSERT,DELETE
trigger_type: INSERT,UPDATE
trigger_type: DELETE,UPDATE
trigger_type: INSERT,DELETE,UPDATE
```

#### **INSERT**

Monitors for the insertion of a row in the database table. This is the default.

**DELETE**

Monitors for the deletion of a row in the database table.

**UPDATE**

Monitors for an update to any of the rows in the database table.

**INSERT,DELETE**

Monitors for the insertion or deletion of a row in the database table.

**Note:** This argument applies to Oracle and SQL Server only.

**INSERT,UPDATE**

Monitors for the insertion of a row or an update to any of the rows in the database table.

**Note:** This argument applies to Oracle and SQL Server only.

**DELETE,UPDATE**

Monitors for the deletion of a row or an update to any of the rows in the database table.

**Note:** This argument applies to Oracle and SQL Server only.

**INSERT,DELETE,UPDATE**

Monitors for the insertion of a row, deletion of a row, or an update to any of the rows in the database table.

**Note:** This argument applies to Oracle and SQL Server only.

**Notes:**

- Depending on the dbtype attribute, CA Workload Automation AE generates the proper syntax for that database type using the values specified in the trigger\_type attribute.
- Each Database Trigger job creates a trigger on the database. We recommend that you speak to your database administrator before creating a Database Trigger job.

### **Example: Specify a Trigger Type for Oracle**

This example monitors the Inventory\_List table for an added row. The job runs on the dbagent agent computer and connects to the ORDERS database. When a row is inserted, the job completes.

```
insert_job: dbtrig_oracle
job_type: DBTRIG
machine: dbagent
tablename: Inventory_List
dbtype: Oracle
trigger_type: INSERT
owner: sys@orcl
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

### **Example: Specify a Trigger Type for SQL Server**

This example monitors the stores table for an added row or a deleted row. The job runs under the sa user, who owns the table and is authorized to create triggers on the database or schema the table belongs to. The job remains in a RUNNING state waiting for an added or deleted row. When a row is either added or deleted, the job completes.

```
insert_job: dbtrig1
job_type: DBTRIG
machine: DB_agent
trigger_type: DELETE,INSERT
tablename: stores
dbtype: MSSQL
owner: sa@myhost
connect_string:"jdbc:sqlserver://myhost:1433;DatabaseName=pubs"
```

### **Example: Specify a Trigger Type for IBM DB2**

This example monitors the STAFF table for changes. The job connects to the SAMPLE database and runs under the entadm user, who is authorized to create triggers on the database or schema the table belongs to. The job completes when the table has changed.

```
insert_job: dbtrig_db2
job_type: DBTRIG
machine: dbagent
dbtype: DB2
owner: entadm@myhost
connect_string:"jdbc:db2://172.31.255.255:50000/SAMPLE"
tablename: STAFF
dbtype: DB2
trigger_type: UPDATE
```

# ulimit Attribute—Specify UNIX Resource Limits

The ulimit attribute modifies the shell's resource usage limit on a UNIX computer.

## Supported Job Type

This attribute is optional for the [Command \(CMD\) job type](#) (see page 204) on UNIX.

## Syntax

This attribute has the following format:

```
ulimit: resource_type="soft_value,hard_value"
[, resource_type="soft_value,hard_value"]
```

### *resource\_type*

Specifies the resource type. The following options are available:

- c—Specifies the core file size in blocks.
- d—Specifies the data segment size in kilobytes (KB).
- f—Specifies the maximum file size in blocks.
- m—Specifies the process virtual size in kilobytes (KB).
- n—Specifies the number of files.
- s—Specifies the stack size in kilobytes (KB).
- t—Specifies the CPU time in seconds.

### *soft\_value*

Defines the usage (soft) limit for the specified resource type. The soft limit can be increased up to the specified hard limit and can be changed without root authority.

**Limits:** Any numeric digits or you can specify **unlimited**; the value must be less than or equal to the hard limit

### *hard\_value*

Defines the maximum usage (hard) limit for the specified resource type. The hard limit can only be increased by the root user. Any user ID can decrease a hard limit.

**Limits:** Any numeric digits or you can specify **unlimited**; the value must be greater than or equal to the soft limit

**Note:** If the job runs under a non-root user ID and the value specified in the job definition is greater than the current hard limit on the UNIX system, the hard limit will not be increased.

**Notes:**

- To find out the lower and upper limits of your operating system, contact your system administrator.
- You can specify multiple *resource\_type*=*soft\_value,hard\_value* combinations. Separate each combination with a comma.
- You can use multiple entries of ulimit to define different sets of resource combinations. For example, you can specify two sets of combinations using two ulimit attributes as follows:

```
ulimit: c="100,200", s="250,300"
ulimit: t="unlimited,4000", m="3332, unlimited"
```
- Each *resource\_type*=*soft\_value,hard\_value* combination can be up to 1024 characters.
- Each entry of ulimit can be up to 4096 characters.
- The ulimit values are applied before the user profile is sourced. Therefore, the values can be overridden by the /etc/profile script, the .profile script, or any other user login scripts.
- This attribute is ignored for CMD jobs on Windows.

**Example: Specify Multiple ulimit Values**

This example runs the procrun.sh script on the unixagent computer. The job modifies the following resource limits on the UNIX computer:

- The core file size limit is 100 KB (soft limit). The size can be increased to 200 KB (hard limit).
- The stack size limit is 250 KB (soft limit). The size can be increased to 300 KB (hard limit).
- The CPU time can be up to 4000 seconds.
- The process virtual size limit is 3332 KB (soft limit). The size can be increased to an unlimited value.

```
insert_job: cmd_job
job_type: CMD
machine: unixagent
command: /u1/procrun.sh
ulimit: c="100,200", s="250,300", t="unlimited,4000", m="3332, unlimited"
```

**Note:** The resource limits are modified for the current job definition only. When you run another job, the default values will be used.

## upper\_boundary Attribute—Define the End of the Range to be Searched

The upper\_boundary attribute defines the end of the range to be searched in a text file.

This attribute is used with the text\_file\_mode attribute, which specifies the search mode (line, regular expression, or date and time).

### Supported Job Type

This attribute is optional for the [Text File Reading and Monitoring \(OMTF\) job type](#) (see page 274).

To use this attribute, you must specify NOW for the monitor\_mode attribute. This attribute does not apply when the monitor mode is set to WAIT or CONTINUOUS.

### Syntax

This attribute has the following format:

upper\_boundary: *end\_range*  
*end\_range*

Defines the end of the range to be searched. The format of this value depends on the text file search mode. The formatting options are as follows:

- Numeric—Specify a numeric value if the search mode is LINE.
- Regular expression—Specify a regular expression if the search mode is REGEX.
- Date and time—Specify date and time values if the search mode is DATETIME.  
Define the date and time using the format specified by the time format.

**Limits:** Up to 256 characters; case-sensitive

**Note:** You cannot specify the TO operand when using the WAITMODE statement with the WAIT value.

### Notes:

- If you do not specify the text\_file\_mode attribute in the job definition, the search mode is LINE by default.
- If you do not define the upper\_boundary attribute in the job definition, the job searches to the last line of the text file.

- When the search mode is REGEX, you can specify a regular expression in the upper\_boundary attribute. To compose a regular expression, follow the rules for Java class java.util.regex.Pattern. You can find these rules using a Google search for java pattern.
- You can use escape sequences in the regular expression. The following characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\).
  - \—back slash
  - |—vertical bar
  - (—opening parenthesis
  - )—closing parenthesis
  - [—opening bracket
  - {—opening brace
  - ^—caret (circumflex)
  - \$—dollar sign
  - \*—asterisk
  - +—plus sign
  - ?—question mark
  - .—period

For example, to match the characters \*.\* literally, specify \\*\.\\* in your regular expression. The backward slashes escape the characters' special meanings.

#### [Example: Define the End of a Range When the Search Mode is Line](#)

This example searches the c:\ca\log file in line mode. The job starts searching the content from line 143 of the file. The upper boundary is not defined, so the job searches to the last line of the file. The job completes successfully if the ERROR MESSAGE string is found.

```
insert_job: omtf_line
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ERROR MESSAGE
text_file_mode: LINE
lower_boundary: 143
monitor_mode: NOW
```

**Example: Define the End of a Range When the Search Mode is RegEx**

This example searches the c:\ca\log file in regular expression mode. The lower boundary is not defined, so the job searches the content from the first line of the file to the upper boundary (a line that contains the word service). The job completes successfully if the ARCHIVE string is found.

```
insert_job: omtf_regex
job_type: OMTF
machine: monagt
text_file_name: "c:\ca\log"
text_file_filter: ARCHIVE
text_file_mode: REGEX
upper_boundary: service
monitor_mode: NOW
```

**Example: Define the End of a Range When the Search Mode is DateTime**

This example searches the /export/home/logs/transmitter.log file in date and time mode. The job searches the content between May 20, 2010 at midnight and May 27, 2010 at 11:59 p.m. The date and time values are defined using the format specified in the time\_format attribute. The job completes successfully if the transmitted string is found.

```
insert_job: omtf_timedate
job_type: OMTF
machine: monagt
text_file_name: /export/home/logs/transmitter.log
text_file_filter: transmitted
text_file_mode: DATETIME
lower_boundary: "Thu May 20 00:00:00.000 EDT 2010"
upper_boundary: "Thu May 27 23:59:59.999 EDT 2010"
time_format: "EEE MMM dd HH:mm:ss.SSS zzz yyyy"
time_position: 12
monitor_mode: NOW
```

## upper\_boundary Attribute—Define the Maximum CPU or Disk Usage to Monitor (OMCPU and OMD Jobs)

The upper\_boundary attribute defines the maximum CPU or disk usage to monitor for.

### Supported Job Types

This attribute is required for the following job types if WAIT or CONTINUOUS is specified for the monitor\_mode attribute:

- [CPU Monitoring \(OMCPU\)](#) (see page 206)
- [Disk Monitoring \(OMD\)](#) (see page 213)

The lower\_boundary attribute does not apply when the monitor mode is NOW.

### Syntax

This attribute has the following format:

- To specify the upper boundary for CPU usage:

    upper\_boundary: *max\_percent*  
    *max\_percent*

        Defines the maximum amount of CPU usage to monitor for in percent.

**Default:** 100

**Limits:** 0-100

- To specify the upper boundary for disk usage:

    upper\_boundary: *max\_value*  
    *max\_value*

        Defines the maximum amount of disk usage to monitor for. The disk\_format attribute specifies the unit for this value.

**Limits:**

- 0-100 (This limit applies when the disk format is PERCENT)
- Up to 20 characters (This limit applies for all other disk formats.)

**Example:** 35000000

### Notes:

- If you specify the upper\_boundary attribute without the lower\_boundary attribute, the range is between zero and the upper boundary.
- The upper\_boundary attribute value must be greater than the lower\_boundary attribute value.

**Example: Specify an Upper Boundary of 75 Percent for Monitoring Available CPU**

This example continuously monitors the CPU available on the winagt computer. An alert is written to the scheduler log file each time the available CPU is less than 25 percent or greater than 75 percent, and the CPU usage changes by more than 10 percent. If the change in value is less than or equal to 10, the job does not register a change.

```
insert_job: omcpu_job
job_type: OMCPU
machine: winagt
cpu_usage: FREE
lower_boundary: 25
upper_boundary: 75
inside_range: FALSE
no_change: 10
monitor_mode: CONTINUOUS
```

**Example: Specify an Upper Boundary of 10 GB for Monitoring Available Disk Space**

This example monitors /export/home for available space. The job completes when the available space is between 8 and 10 gigabytes (GB), and the available space changes by more than 15 percent. If the change in value is less than or equal to 15, the job does not register a change.

```
insert_job: omd_unix
job_type: OMD
machine: unixagt
disk_drive: /export/home
disk_space: FREE
disk_format: GB
lower_boundary: 8
upper_boundary: 10
inside_range: TRUE
no_change: 15
```

## URL Attribute—Specify the URL of the JMX Server

The URL attribute specifies the URL to connect to the JMX server in a JMX job.

### Supported Job Types

This attribute is required for the following job types:

- [JMX-MBean Attribute Get \(JMXMLG\)](#) (see page 232)
- [JMX-MBean Attribute Set \(JMXMLAS\)](#) (see page 233)
- [JMX-MBean Create Instance \(JMXMLC\)](#) (see page 235)
- [JMX-MBean Operation \(JMXMLOP\)](#) (see page 237)
- [JMX-MBean Remove Instance \(JMXMLREM\)](#) (see page 239)
- [JMX-MBean Subscribe \(JMXMLSUB\)](#) (see page 240)

### Syntax

This attribute has the following format:

URL: "*url*"

**"*url*"**

Specifies the URL to connect to the JMX server using an RMI connector. The format is the following:

"service:jmx:rmi:///jndi/rmi:///*hostName*:*portNum*/*jmxServerName*"

***hostName***

Specifies the host name or IP address of the JMX server.

***portNum***

Specifies the port number of the JMX server.

***jmxServerName***

Specifies the name of the JMX server.

**Limits:** Up to 256 characters; case-sensitive

**Example:** "service:jmx:rmi:///jndi/rmi://localhost:9999/server"

**Example: Query a JMX Server for the Value of an MBean Attribute**

Suppose that you want to know the value of the cachesize attribute for the Config MBean. The URL for the JMX server is service:jmx:rmi:///jndi/rmi://localhost:9999/server, where localhost is the host name and 9999 is the port number.

```
insert_job: query
job_type: JMXTAG
machine: appagent
URL: "service:jmx:rmi:///jndi/rmi://localhost:9999/server"
mbean_name: "DefaultDomain:index=1,type=Config"
mbean_attr: cachesize
```

## use\_topic Attribute—Publish or Subscribe to a Topic or Queue

The use\_topic attribute specifies whether the job sends messages to a topic or queue in a JMS Subscribe job or publishes messages to a topic or queue in a JMS Publish job.

**Note:** If you do not specify the use\_topic attribute in your job definition, the job sends or publishes messages to a topic (the default).

### Supported Job Types

This attribute is optional for the following job types:

- [JMS Publish \(JMSPUB\)](#) (see page 228)
- [JMS Subscribe \(JMSSUB\)](#) (see page 230)

### Syntax

This attribute has the following format:

`use_topic: TRUE | FALSE`

#### **TRUE**

Sends or publishes messages to a topic.

#### **FALSE**

Sends or publishes messages to a queue. This is the default.

#### **Example: Publish a Message to a Queue**

This example publishes the message "this is my message" to the queue named Queue. The Java class of the message is String. The initial context factory supplied by the JNDI service provider is com.ibm.websphere.naming.WsnInitialContextFactory. The service provider's URL is iiop://172.24.0.0:2809, where 172.24.0.0 is the IP address of the WebSphere MQ server and 2809 is the ORB port. The job uses the cyberuser JNDI user name to gain access to the connection factory named ConnectionFactory.

```
insert_job: publish
job_type: JMSPUB
machine: appagent
initial_context_factory: com.ibm.websphere.naming.WsnInitialContextFactory
provider_url: "iiop://172.24.0.0:2809"
connection_factory: ConnectionFactory
destination_name: Queue
use_topic: FALSE
message_class: String
j2ee_user: cyberuser
j2ee_parameter: java.lang.String="this is my message"
```

**Note:** The agent does not support JMS messaging on IBM WebSphere. If you have IBM WebSphere MQ, your agent administrator can set up the agent plug-in to run JMS Publish and JMS Subscribe for JMS queues. JMS topics are not supported on IBM WebSphere MQ.

## **user\_role Attribute—Specify Database Resource Location**

The user\_role attribute specifies the Oracle database user type for a database job.

### **Supported Job Types**

This attribute is optional for the following job types:

- [Database Monitor \(DBMON\)](#) (see page 207)
- [Database Stored Procedure \(DBPROC\)](#) (see page 209)
- [Database Trigger \(DBTRIG\)](#) (see page 211)
- [SQL](#) (see page 272)

### **Syntax**

This attribute has the following format:

```
user_role: user_type
```

***user\_type***

Specifies the role of the Oracle user to log in as. The user role must be defined in the Oracle database.

**Limits:** Up to 30 characters; case-sensitive

**Notes:**

- If you omit this attribute from the job definition, the job runs with normal privileges.
- In Oracle, a user with sufficient authority can log in with different system privileges. For example, if a job requires sysdba privileges, enter **as sysdba** in this field.
- Your agent administrator can specify a default user type for all database jobs by setting the db.default.userType parameter in the agent's agentparm.txt file.
- The user\_role attribute overrides the default user type specified in the agent's agentparm.txt.

**Example: Specify a Database User Type**

This example specifies a database user named dbuser1, who is logged in with sysdba privileges. The job connects to the Oracle database named ORDERS, and the output is stored in the job's spool file.

```
insert_job: QRY1
job_type: SQL
machine: dbagent
owner: dbuser1@orcl
user_role: as sysdba
sql_command: SELECT * from NEWORDS
connect_string: "jdbc:oracle:thin:@172.31.255.255:1433:ORDERS"
```

## watch\_file Attribute—Specify a File to Monitor

The watch\_file attribute specifies the path to and name of a file to monitor.

### Supported Job Types

This attribute is required for the following job types:

- [File Trigger \(FT\)](#) (see page 217)
- [File Watcher \(FW\)](#) (see page 219)

### UNIX and Windows Syntax

This attribute has the following format:

```
watch_file: file
file
```

Specifies the path to and name of one or more files to monitor. To specify multiple files, use the asterisk (\*) and question mark (?) wildcard characters. Use \* for any number of characters and ? for any single character. Do not use the \* and ? in the path.

**Limits:** Up to 255 characters; case-sensitive

### UNIX Notes:

- You can specify variables exported from the profile script or global variables in the name.
- Wildcard characters are expanded according to the wildcard expansion rules of the Bourne Shell.

**Windows Notes:**

- You can specify system environment variables, job profile environment variables, and global variables in the name. If the variable is referenced in the job profile, we recommend that you enclose the variable in braces (for example, \${PATH}). Use the expression \$\$*global\_name* for global variables.
- When specifying drive letters in job definitions, you must escape the colon with quotation marks or backslashes. For example, C:\\tmp or "C:\\tmp" is valid; C:\\tmp is not.

**Note:** An application that generates the monitored file may crash, or a communication link may be interrupted after the minimum size file is written. In these situations, CA Workload Automation AE may incorrectly determine that the file is complete when it is not. If you have the permissions to control the application that generates the monitored file, you can do the following to avoid an incorrect result:

- Instruct the application to create a separate, zero-length file when it finishes writing the monitored file.
- Set the job to monitor for the completion of the zero-length file.

If the job detects the zero-length file, the job is complete.

## File Trigger Notes

When you define a File Trigger (FT) job, consider the following points:

Scan intervals:

File Trigger jobs monitor file activity using a polling interval, which is every 30 seconds by default. File Trigger jobs do not detect multiple updates during the polling interval. They also do not detect changes that cancel each other out.

For example, if the job monitors for updates to a file, and the file is updated twice during the polling interval, the trigger occurs only once for the two updates. If the job monitors for the creation of a file, and the file is created and deleted during the polling interval, the trigger does not occur. Because the file did not exist when the directory was polled, the job does not detect the file creation and deletion.

**Note:** To change the number of seconds between scans, specify the watch\_no\_change attribute in your job definition.

#### Specifying wildcards in the file name:

- You can use wildcards in the file name value. The asterisk (\*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.

In the following example, the agent monitors for any file in the /usr/data directory:

```
watch_file: "/usr/data/*"
```

In the following example, the agent monitors for any file in the /usr/data directory that has a file name with four characters and any extension:

```
watch_file: /usr/data/????.*
```

- You cannot use a wildcard as a prefix or within the directory names in the file path.
- When using a wildcard immediately after a forward slash (/), enclose the path in quotation marks. The quotation marks prevent part of the path from being recognized as a comment line. For example, this statement might not produce the desired results:

```
watch_file: /usr/data/*
```

To include the wildcard as part of the path, enclose the path in quotation marks, as follows:

```
watch_file: "/usr/data/*"
```

#### Specifying a file on a remote Windows computer:

You can specify UNC (Universal Naming Convention) names. A UNC name is the name of a file or other resource that begins with two backslashes (\\\), indicating that it exists on a remote computer.

#### Specifying an i5/OS object:

You can use generic names to specify library names and object names on the i5/OS system. A generic name starts with characters that are part of a valid name and ends with an asterisk (\*). The asterisk denotes any number of characters and can only be placed at the end of a generic name.

The following watch\_file attribute prompts the agent to monitor for the creation of any file object in the QSYS file system with a file name that matches the PAY\* generic name:

```
watch_file: "LIB/PAY*/*FILE"
```

**Note:** A name cannot contain a single asterisk and no other characters. To specify all possible names, use the special value \*ALL.

## Examples: Specifying UNIX files using the watch\_file Attribute

The following examples specify UNIX files to monitor:

**Note:** The following examples are File Trigger jobs. If you are using a legacy agent, you can define these examples as File Watcher jobs by modifying the job\_type attribute to **job\_type: FW**.

- This example monitors the /tmp/batch.input file.

```
insert_job: ft_unix1
job_type: FT
machine: unixagt
watch_file: /tmp/batch.input
```

- This example monitors a file whose name has been assigned to the global variable file\_1.

```
insert_job: ft_unix2
job_type: FT
machine: unixagt
watch_file: ${file_1}
```

- This example monitors files with the format, /usr/reports/Augustxx.out (such as /usr/reports/August01.out and /usr/reports/August02.out).

```
insert_job: ft_unix3
job_type: FT
machine: unixagt
watch_file: /usr/reports/August???.out
```

- This example monitors the /tmp directory for files with names that begin with "pay".

```
insert_job: ft_unix4
job_type: FT
machine: unixagt
watch_file: /tmp/pay*
```

## Examples: Specifying Windows files using the watch\_file Attribute

The following examples specify Windows files to monitor:

**Note:** The following examples are File Trigger jobs. If you are using a legacy agent, you can define these examples as File Watcher jobs by modifying the job\_type attribute to **job\_type: FW**.

- This example monitors the C:\tmp\BATCH.IN file.

```
insert_job: ft_win1
job_type: FT
machine: winagt
watch_file: "C:\tmp\BATCH.IN"
```

- This example monitors a file whose name has been assigned to the global variable file\_1.

```
insert_job: ft_win2
job_type: FT
machine: winagt
watch_file: ${file_1}
```

- This example monitors files with the format, c:\reports\Augustxx.out (such as C:\reports\August01.out and C:\reports\August02.out).

```
insert_job: ft_win3
job_type: FT
machine: winagt
watch_file: "c:\reports\August???.out"
```

- This example monitors the temp directory for files with names that begin with "pay".

```
insert_job: ft_win4
job_type: FT
machine: winagt
watch_file: "c:\temp\pay*"
```

## Examples: Specifying i5/OS files using the watch\_file Attribute

The following examples specify i5/OS files to monitor:

- This example monitors a daily update to a payroll file object on an i5/OS computer. When the file is updated, the job completes.

```
insert_job: ft_i5os1
job_type: FT
machine: i5agt
watch_file: /QSYS.LIB/LIBRARY.LIB/DEPT.FILE/PAYROLL.MB
watch_file_type: UPDATE
```

- This example monitors for files that are created in the /home/cybesp/ directory in the root file system. The job completes if a file is created with a file name that matches the following criteria:

- Starts with PID
  - Ends with four characters
  - Has any extension.

```
insert_job: ft_i5os2
job_type: FT
machine: i5agt
watch_file: /home/cybesp/PID????.*
```

watch\_file\_type: CREATE

- This example uses a generic name to specify a QSYS file object in an i5/OS computer. The job completes when any file object with a file name that starts with PAY and ends with any characters is created in the QSYS file system. Note that the file name is enclosed in quotation marks so that the text following /\* is not interpreted as a comment.

```
insert_job: ft_i5os3
job_type: FT
machine: i5agt
watch_file: "LIB/PAY/*FILE"
watch_file_type: CREATE
```

- This example monitors the size of a QSYS file object in an i5/OS computer. The job completes when the EXPOBJ file object in the QSYS file system increases by 10 bytes or more. Since no member is specified in the file name, the job monitors the entire object size. The entire object size includes the size of the file object itself and the total size of the file object's members.

```
insert_job: ft_i5os3
job_type: FT
machine: i5agt
watch_file: "LIB/EXPOBJ/*FILE"
watch_file_type: EXPAND
watch_file_type: SIZE
watch_file_change_value: 10
```

**Notes:**

- The file name is enclosed in single quotation marks so that the text following /\* is not interpreted as a comment.
  - To monitor only the total size of the file object's members, specify \*ALL as the member name.

## watch\_file\_change\_type Attribute—Specify the Type of Change in File Size

The `watch_file_change_type` attribute specifies the type of change to detect when monitoring a file for an increase or decrease in size.

### Notes:

- If you do not specify the `watch_file_change_type` attribute in your job definition, the job monitors the file to reach a specified size (the default).
- To use this attribute, you must include the `watch_file_change_value` attribute in your job definition. These attributes correspond to each other.

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217) as follows:

- If EXPAND or SHRINK is specified for the `watch_file_type` attribute, you can specify DELTA, PERCENT, or SIZE.
- If CREATE is specified for the `watch_file_type` attribute, you can specify SIZE.

### Syntax

This attribute has the following format:

`watch_file_change_type: DELTA | PERCENT | SIZE`

#### **DELTA**

Specifies that the job monitors for a change in the file size. The unit is bytes.

#### **PERCENT**

Specifies that the job monitors for a change in the file size by percentage.

#### **SIZE**

Specifies that the job monitors the file to reach a specified size. The unit is bytes. This is the default.

**Note:** Some file systems will create files with extra space in anticipation of further file expansion requests. Therefore, specifying exact byte counts will not always work.

#### **Example: Monitor for an Increase in File Size by a Certain Amount**

In this example, the unixagt agent monitors the record file in the credit directory. If the record file expands in size by 200,000 bytes or more, the job completes.

```
insert_job: ft_unix_delta
job_type: FT
machine: unixagt
watch_file: /credit/record
watch_file_type: EXPAND
watch_file_change_type: DELTA
watch_file_change_value: 200000
```

#### **Example: Monitor for a Decrease in File Size to a Certain Amount**

In this example, the unixagt agent monitors the distribute file in the /cash/items directory. If the file size shrinks to 1000 bytes or smaller, the job completes.

```
insert_job: ft_unix_shrink
job_type: FT
machine: unixagt
watch_file: /cash/items/distribute
watch_file_type: SHRINK
watch_file_change_type: SIZE
watch_file_change_value: 1000
```

#### **Example: Monitor for a Decrease in File Size by a Certain Percentage**

In this example, the winagt agent monitors the test file in the c:\amount directory. When the file shrinks in size by 65% or more, the job completes.

```
insert_job: ft_win_percent
job_type: FT
machine: winagt
watch_file: "c:\amount\test"
watch_file_type: SHRINK
watch_file_change_type: PERCENT
watch_file_change_value: 65
```

## watch\_file\_change\_value Attribute—Specify the Change in File Size

The watch\_file\_change\_value attribute specifies the amount of change or the limit in size to detect when monitoring a file for an increase or decrease in size.

**Note:** To use this attribute, you must include the watch\_file\_change\_type attribute in your job definition. The watch\_file\_change\_type specifies the type of unit associated with this value.

### Supported Job Type

This attribute is required for the [File Trigger \(FT\) job type](#) (see page 217) if EXPAND or SHRINK is specified for the watch\_file\_type attribute.

This attribute is optional for the File Trigger job type if watch\_file\_type: CREATE and watch\_file\_change\_type: SIZE are specified.

### Syntax

This attribute has the following format:

```
watch_file_change_value: value
```

**value**

Defines *one* of the following:

- When the file change type is DELTA, this attribute defines the number of bytes the file size must change for the file trigger to occur.
- When the file change type is PERCENT, this attribute defines the percentage the file size must change for the file trigger to occur.
- When the file change type is SIZE, this attribute defines a limit on the file size. If the type of activity is CREATE or EXPAND, the file trigger occurs when the file size is equal to or greater than the specified size. If the type of activity is SHRINK, the file trigger occurs when the file size is equal to or less than the specified size.

**Limits:** Up to 19 digits

**i5/OS:** When monitoring a \*FILE object in QSYS with no member specified in the file name, the job monitors the entire object size. The entire object size includes the size of the file object itself and the total size of the file object's members. To only monitor the total size of the file object's members, specify \*ALL as the member name.

**Notes:**

- The maximum value is 9,223,372,036,854,775,807.
- If watch\_file\_type is set to EXPAND or SHRINK, the value must be greater than or equal to 1.
- If watch\_file\_type is set to SHRINK and watch\_file\_change\_value is set to PERCENT, this value must be between 1 and 100.
- If watch\_file\_type is set to EXPAND and watch\_file\_change\_value is set to PERCENT, this value must be between 1 and 10000.

**Example: Monitor for an Increase in File Size by a Certain Percentage**

This example monitors file test in directory amount. When the file expands in size by 90% or more, the job completes. For instance, suppose that the current file size is 1000 bytes. The file trigger occurs when the file size expands to 1900 bytes or more.

```
insert_job: ft_win_percent
job_type: FT
machine: unixagt
watch_file: /amount/test
watch_file_type: EXPAND
watch_file_change_type: PERCENT
watch_file_change_value: 90
```

## watch\_file\_groupname Attribute—Specify the Group that Owns the File

The `watch_file_groupname` attribute specifies the name of the group that owns the file to be monitored.

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217) on UNIX.

### Syntax

This attribute has the following format:

`watch_file_groupname: group`

#### *group*

Specifies the name of the group that owns the file to be monitored.

**Limits:** Up to 80 characters; case-sensitive; it cannot contain delimiters (such as spaces)

#### Notes:

- Applies only to UNIX jobs and i5/OS jobs (if the file is not a QSYS object).
- If the file is not owned by the specified group, the following occurs:
  - The job continues monitoring if the type of file activity is CREATE.
  - The job completes if the type of file activity is DELETE or NOTEXIST.
  - The job fails if the type of file activity is EXPAND, EXIST, SHRINK, or UPDATE.

### Example: Monitor for the Existence of a File that Is Owned by a Specified UNIX Group

Suppose that you want a job to monitor for the existence of the `/data/payroll.dat` file that is owned by the UNIX group ACCTS:

- If the file exists and the file is owned by the group ACCTS when the job is readied, the job completes immediately.
- If the file exists and the file is not owned by the group ACCTS when the job is readied, the job fails.

```
insert_job: ft_unixgroup
job_type: FT
machine: unixagent
watch_file: /data/payroll.dat
watch_file_type: EXIST
watch_file_groupname: ACCTS
```

## watch\_file\_min\_size Attribute—Specify the Minimum Size that a File Must Reach

The watch\_file\_min\_size attribute specifies the minimum size that the watched file must reach for the job to complete. The watched file must maintain a steady state during the interval specified in the watch\_file attribute.

**Note:** If you do not specify the watch\_file\_min\_size attribute in your job definition, the job completes if the file exists (the default).

### Supported Job Type

This attribute is optional for the [File Watcher \(FW\) job type](#) (see page 219).

### Syntax

This attribute has the following format:

watch\_file\_min\_size: *bytes*  
***bytes***

Specifies the minimum file size (in bytes) that must be reached to cause this job to complete.

**Limits:** Up to 19 digits

**Default:** 0 (The job completes if the file exists.)

**Note:** The maximum value is 9,223,372,036,854,775,807.

### Example: Specify the Minimum File Size

This example monitors the watch\_file.log file on the unixagent computer. The unixagent is a legacy agent. The job completes when the file reaches 10000 bytes and maintains a steady state for 60 seconds.

```
insert_job: fw_job
job_type: FW
machine: unixagent
watch_file: /tmp/watch_file.log
watch_file_min_size: 10000
watch_interval: 60
```

## watch\_file\_owner Attribute—Specify the Owner of the File to be Monitored

The `watch_file_owner` attribute specifies the user ID that owns the file to be monitored.

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217) on UNIX.

### Syntax

This attribute has the following format:

`watch_file_owner: owner`

*owner*

Specifies the user ID that owns the file to be monitored.

**Limits:** Up to 80 characters; case-sensitive; it cannot contain delimiters (such as spaces)

#### Notes:

- Applies only to UNIX jobs and i5/OS jobs (if the file is not a QSYS object).
- If the file is not owned by this group:
  - The job continues monitoring if the type of file activity is CREATE.
  - The job completes if the type of file activity is DELETE or NOTEXIST.
  - The job fails if the type of file activity is EXPAND, EXIST, SHRINK, or UPDATE.

### Example: Monitor a File Owned by a Specific User

This example monitors for the creation of a file named payroll owned by JDOE.

```
insert_job: ft_job
job_type: FT
machine: ftagt
watch_file: /usr/data/payroll
watch_file_type: CREATE
watch_file_owner: JDOE
```

The job completes as follows:

- If the file does not exist when the job starts, the job does not complete until the file is created.
- If the file exists and the owner of the file is JDOE when the job is readied, the job completes immediately.

If the file exists or is created, but the owner of the file is not JDOE, the job does not complete. It waits until all specified criteria are satisfied, including the owner criteria. If the owner of the file changes to JDOE, the job completes.

## watch\_file\_recursive Attribute—Specify Whether to Monitor Subdirectories for File Activity

The `watch_file_recursive` attribute specifies whether the job monitors for file activity in the specified directory only or in the specified directory and all of its subdirectories.

**Note:** If you do not specify the `watch_file_recursive` attribute in your job definition, the job monitors for file activity in the specified directory only (the default).

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217).

### Syntax

This attribute has the following format:

`watch_file_recursive: Y | N`

**Y**

Specifies that the job monitors for file activity in the specified directory and all of its subdirectories.

**N**

Specifies that the job monitors for file activity in the specified directory only. This is the default.

### Example: Monitor for an Update to a File in a Directory and its Subdirectories

This example monitors the `/usr/data/` directory and its subdirectories for the payroll file. When the payroll file is updated in any of the monitored directories, the job completes.

```
insert_job: ft_unix_update
job_type: FT
machine: unixagt
watch_file: /usr/data/payroll
watch_file_type: UPDATE
watch_file_recursive: Y
```

#### **Example: Monitor for Updates to All Files in a Directory and its Subdirectories**

This example monitors all the files in the /usr/data/ directory and its subdirectories. When any file is updated in any of the monitored directories, the job completes.

```
insert_job: ft_unix_update
job_type: FT
machine: unixagt
watch_file: "/usr/data/*"
watch_file_type: UPDATE
watch_file_recursive: Y
```

#### **Example: Monitor a Directory and Its Subdirectories for File Expansion**

This example monitors a file named inventory.txt and all other jobs in that directory and its subdirectories. When the size of the inventory.txt file or any other file in that same directory increases by 10 bytes, the job completes and the output is recorded in the ftjob1\_out.log file.

```
insert_job: ft_job
job_type: FT
machine: ftagt
watch_file: "c:\fwjobs\inventory.txt"
watch_file_type: Expand
watch_file_change_type: DELTA
watch_file_change_value: 10
watch_file_recursive: Y
```

## watch\_file\_type Attribute—Specify the Type of File Activity to Monitor For

The `watch_file_type` attribute specifies the type of file activity that a File Trigger job monitors for.

**Note:** If you do not specify the `watch_file_type` attribute in your job definition, the job monitors for the creation of the file (the default).

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217).

### Syntax

This attribute has the following format:

```
watch_file_type: CREATE | DELETE | EXIST | EXPAND |
 NOTEXIST | SHRINK | UPDATE | GENERATE
```

#### CREATE

Indicates that the file trigger occurs when the file is created. This is the default.

#### **DELETE**

Indicates that the file trigger occurs when the file is deleted.

#### **EXIST**

Indicates that the file trigger occurs if the file exists. If the file does not exist, the job fails.

#### **EXPAND**

Indicates that the file trigger occurs when the file size increases.

**Note:** To monitor for this type of file activity, you must specify the `watch_file_change_type` attribute in your job definition.

#### **NOTEXIST**

Indicates that the file trigger occurs if the file does not exist. If the file exists, the job fails.

#### **SHRINK**

Indicates that the file trigger occurs when the file size decreases.

**Note:** To monitor for this type of file activity, you must specify the `watch_file_change_type` attribute in your job definition.

#### **UPDATE**

Indicates that the file trigger occurs when the file is updated.

#### GENERATE

Indicates that the file trigger occurs when the file remains unchanged for the amount of time specified in the `watch_no_change` attribute.

**Note:** For more information and examples about each file activity, see the notes for that type.

## CREATE File Trigger Type Notes

When specifying the CREATE value in the `watch_file_type` attribute, consider the following points:

- If the file name specified in the job definition contains wildcards, the first matching file is selected.
- If the file exists when the job starts, the trigger occurs immediately. If the file does not exist when the job starts, the trigger does not occur until the file is created.
- If the specified directory does not exist or is deleted during monitoring, the job fails.
- If the file owner and group are specified in the job definition, the file trigger occurs only when all specified criteria are satisfied, including the owner and group criteria.

### Example: Monitor the Creation of a File Owned by a Specific User

This example monitors for the creation of a file named `payroll` owned by `JDOE`.

```
insert_job: ft_job
job_type: FT
machine: ftagt
watch_file: /usr/data/payroll
watch_file_type: CREATE
watch_file_owner: JDOE
```

The job completes as follows:

- If the file does not exist when the job starts, the job does not complete until the file is created.
- If the file exists and the owner of the file is `JDOE` when the job is readied, the job completes immediately.
- If the file exists or is created, but the owner of the file is not `JDOE`, the job does not complete. It waits until all specified criteria are satisfied, including the owner criteria. If the owner of the file changes to `JDOE`, the job completes.

## DELETE File Trigger Type Notes

When specifying the DELETE value in the watch\_file\_type attribute, consider the following points:

- If the file does not exist when the job starts, the file trigger occurs immediately.
- If the specified directory does not exist or is deleted during monitoring, the job fails.
- You can specify the continuous attribute with DELETE only when you are using a wildcard (\*) or (?) in the file name value.

### Example: Monitor for the Deletion of Files That Have Names Beginning with Pay

This example continuously monitors the /usr/data/ directory for files that have names beginning with pay. When all files that have a name beginning with pay are deleted, the job completes successfully.

```
insert_job: ftjob
job_type: FT
machine: ftagt
watch_file: /usr/data/pay*
watch_file_type: DELETE
continuous: Y
```

## EXIST File Trigger Type Notes

When specifying the EXIST value in the watch\_file\_type attribute, consider the following points:

- If the file does not exist, the job fails.
- If the file name specified in the watch\_file attribute contains wildcards, the file with the most recent modification time that matches the criteria is monitored.

### Example: Check that a File Exists in a Directory

In this example, the winagt agent monitors for file money in directory c:\bank\account. If the money file exists in that directory, the job completes. If the money file does not exist in that directory, the job fails.

```
insert_job: ft_win_exist
job_type: FT
machine: winagt
watch_file: "c:\bank\account\money"
watch_file_type: EXIST
```

**Example: Check that a File Exists in a Directory or Its Subdirectories**

In this example, the winagt agent monitors for the file named money in directory c:\bank\account and all c:\bank\account subdirectories. If the file exists in any of the monitored directories, the job completes. If the money file does not exist in any of the monitored directories, the job fails.

```
insert_job: ft_win_recursive
job_type: FT
machine: winagt
watch_file: "c:\bank\account\money"
watch_file_type: EXIST
watch_file_recursive: Y
```

## EXPAND File Trigger Type Notes

When specifying the EXPAND value in the watch\_file\_type attribute, consider the following points:

Job failure scenarios:

- If the file does not exist when the file trigger is set, the job fails.
- Deleting the monitored file causes the job to fail.

Monitoring for an increase by size, percent, or delta:

- The following attributes specify that the file is monitored until its size increases to *n* or greater than *n*:

```
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: n
```

If the file size at the initial monitor time is equal to or greater than *n*, the job completes immediately.

- The following attributes specify that the file is monitored for a change in size by percent:

```
watch_file_type: EXPAND
watch_file_change_type: PERCENT
```

If the file size is 0, the trigger occurs when the file size increases.

- The following attributes specify that the file is monitored until its size increases by *n* (change in size):

```
watch_file_type: EXPAND
watch_file_change_type: DELTA
watch_file_change_value: n
```

Specifying wildcards in the file name:

If the file name specified in the job definition contains wildcards, and more than one matching file exists, the file with the most current modification time that matches the criteria is monitored for the duration of the trigger. The only way to determine which file is being monitored is to view the debug output.

Specifying the owner or group in the job definition:

If the file owner or group is specified in the job definition, the file trigger occurs only when all specified criteria are satisfied, including the owner or group criteria. If the owner or group does not match, the job fails.

Continuously monitoring for an increase in file size:

The following attributes specify that the file size is monitored continuously for an increase to *n*:

```
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: n
continuous: y
```

If the initial file size is greater than *n*, the *n* value becomes the new upper limit, and the file continues to be monitored. For example, this table shows when the trigger will occur if the job monitors for an increase in size by 10 bytes:

| The file size when the agent scans the directory | Does the file trigger occur?                                                                                     |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| 15 bytes                                         | Yes. The file is bigger than 10 bytes when the directory is first scanned. 15 bytes becomes the new upper limit. |
| 20 bytes                                         | Yes. 20 bytes exceeds the upper limit of 15 bytes. 20 bytes becomes the new upper limit.                         |
| 35 bytes                                         | Yes. 35 bytes becomes the new upper limit.                                                                       |
| 30 bytes                                         | No. The upper limit is now 35 bytes. The size needs to exceed 35 bytes for the trigger to occur.                 |
| 40 bytes                                         | Yes. 40 bytes exceeds the upper limit of 35 bytes. 40 bytes becomes the new upper limit.                         |
| 45 bytes                                         | Yes. 45 bytes becomes the new upper limit.                                                                       |
| 60 bytes                                         | Yes. 60 bytes becomes the new upper limit.                                                                       |
| 50 bytes                                         | No. The upper limit is now 60 bytes. The size needs to exceed 60 bytes for the trigger to occur.                 |

| The file size when the agent scans the directory | Does the file trigger occur?                                                                |
|--------------------------------------------------|---------------------------------------------------------------------------------------------|
| 65 bytes                                         | Yes. 65 bytes exceeds the upper limit of 60 bytes.<br>65 bytes becomes the new upper limit. |

#### Example: Monitor for an Increase in File Size to a Certain Amount

This example monitors the test file in the data directory. When that file size reaches 100 bytes or more, the job completes.

```
insert_job: ft_unix_size
job_type: FT
machine: unixagt
watch_file: /data/test
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: 100
```

#### Example: Monitor for an Increase in File Size by a Certain Amount

In this example, the unixagt agent monitors the record file in the credit directory. If the record file expands in size by 200,000 bytes or more, the job completes.

```
insert_job: ft_unix_delta
job_type: FT
machine: unixagt
watch_file: /credit/record
watch_file_type: EXPAND
watch_file_change_type: DELTA
watch_file_change_value: 200000
```

#### Example: Monitor a File for Size Increase

This example monitors a file named inventory.txt in the c:\fwjobs directory. When the size of the file increases to 1000 bytes, the job completes.

```
insert_job: ftjob
job_type: FW
machine: ftagt
watch_file: "c:\fwjobs\inventory.txt"
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: 1000
```

## NOTEXIST File Trigger Type Notes

When specifying the NOTEXIST value in the watch\_file\_type attribute, consider the following points:

- If the file does not exist, the job completes successfully.
- If the specified directory does not exist or is deleted during monitoring, the job fails.
- If the file name specified in the watch\_file attribute contains wildcards, the file with the most recent modification time that matches the criteria is monitored.

### Example: Check that a File Does Not Exist in a Directory

In this example, the unixagt agent checks for the file named vacation in directory /start/term/. If the vacation file does not exist in that directory, the job completes. If the vacation file exists in that directory, the job fails.

```
insert_job: ft_unix_notexist
job_type: FT
machine: unixagt
watch_file: /start/term/vacation
watch_file_type: NOTEXIST
```

## SHRINK File Trigger Type Notes

When specifying the SHRINK value in the watch\_file\_type attribute, consider the following points:

Job failure scenarios:

- If the file does not exist when the file trigger is set, the job fails.
- Deleting the monitored file causes the job to fail.

Monitoring for a decrease by size or delta:

- The following attributes specify that the file is monitored until its size decreases to *n* or less than *n*:

```
watch_file_type: SHRINK
watch_file_change_type: SIZE
watch_file_change_value: n
```

If the file size at the initial monitor time is equal to or less than *n*, the job completes immediately.

- The following attributes specify that the file is monitored until its size decreases by *n* (change in size):

```
watch_file_type: SHRINK
watch_file_change_type: DELTA
watch_file_change_value: n
```

When the file size decreases by *n*, the job completes. If the file size at the initial monitor time is less than *n*, the size cannot shrink by *n*. Therefore, the job completes when the file size reaches 0. For example, suppose that the monitored file has an initial file size of 5 bytes and the following attributes are defined:

```
watch_file_type: SHRINK
watch_file_change_type: DELTA
watch_file_change_value: 10
```

The file size (5) is less than the change value (10), so the job cannot shrink by 10. The job completes when the file reaches 0 bytes.

Continuously monitoring for a decrease in file size:

The following attributes specify that the file size is monitored continuously until its size decreases to  $n$  or less than  $n$ :

```
watch_file_type: SHRINK
watch_file_change_type: SIZE
watch_file_change_value: n
continuous: Y
```

If the file size at the initial monitor time is less than  $n$ , the  $n$  value becomes the new lower limit and the file continues to be monitored. The file triggers each time the size decreases to  $n$  or less than  $n$ , and the lower limit is set to the new value. For example, suppose that the following attributes are defined:

```
watch_file_type: SHRINK
watch_file_change_type: SIZE
watch_file_change_value: 15
continuous: Y
```

The following table shows when the trigger will occur:

| The file size when the agent scans the directory | Does the file trigger occur?                                                                                                         |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| 10 bytes                                         | Yes. The file is less than $n$ (15 bytes) when the directory is first scanned. 10 bytes becomes the new lower limit.                 |
| 12 bytes                                         | No. 12 bytes exceeds the lower limit of 10 bytes. The file size needs to be equal to or less than 10 bytes for the trigger to occur. |
| 5 bytes                                          | Yes. 5 bytes becomes the new lower limit.                                                                                            |
| 0 bytes                                          | No. The job continues to run until you manually complete it.                                                                         |

Continuously monitoring for a change in file size:

The following attributes specify that the file size is monitored continuously for shrinkage by *n* (change in size):

```
watch_file_type: SHRINK
watch_file_change_type: DELTA
watch_file_change_value: n
continuous: Y
```

For example, suppose that the following attributes are defined:

```
watch_file_type: SHRINK
watch_file_change_type: DELTA
watch_file_change_value: 10
continuous: Y
```

This table shows when the trigger will occur:

| The file size when the agent scans the directory | Does the file trigger occur?                                                                                  |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| 25 bytes                                         | No. There is no change in file size when the directory is first scanned.                                      |
| 15 bytes                                         | Yes. The change in file size is 10 bytes. The trigger continues to monitor the file for shrinkage to 5 bytes. |
| 5 bytes                                          | Yes. The change in file size is 10 bytes. The job continues to run until you manually complete it.            |
| 0 bytes                                          | No. The job continues to run until you manually complete it.                                                  |

Specifying wildcards in the file name:

If the file name specified in the job definition contains wildcards, and more than one matching file exists, the file with the most current modification time that matches the criteria is monitored for the duration of the trigger. The only way to determine which file is being monitored is to view the debug output.

Specifying the owner or group in the job definition:

If the file owner or group is specified in the job definition, the file trigger occurs only when all specified criteria are satisfied, including the owner or group criteria. If the owner or group does not match, the job fails.

#### **Example: Monitor for Decrease in File Size to a Certain Amount**

In this example, the unixagt agent monitors the distribute file in the /cash/items directory. If the file size shrinks to 1000 bytes or smaller, the job completes.

```
insert_job: ft_unix_shrink
job_type: FT
machine: unixagt
watch_file: /cash/items/distribute
watch_file_type: SHRINK
watch_file_change_type: SIZE
watch_file_change_value: 1000
```

#### **Example: Monitor for Decrease in File Size by a Certain Amount**

In this example, the winagt agent monitors the cash file in the c:\cost directory. When the file shrinks by 10 bytes, the job completes.

```
insert_job: ft_win_shrink
job_type: FT
machine: winagt
watch_file: "c:\cost\cash"
watch_file_type: SHRINK
watch_file_change_type: DELTA
watch_file_change_value: 10
```

#### **Example: Monitor for Decrease in File Size by a Certain Percentage**

In this example, the winagt agent monitors the test file in the c:\amount directory. When the file shrinks in size by 65% or more, the job completes.

```
insert_job: ft_win_percent
job_type: FT
machine: winagt
watch_file: "c:\amount\test"
watch_file_type: SHRINK
watch_file_change_type: PERCENT
watch_file_change_value: 65
```

## UPDATE File Trigger Type Notes

When specifying the UPDATE value in the watch\_file\_type attribute, consider the following points:

Job success scenario:

If the file exists when the job starts, the trigger occurs when the file is updated.

Job failure scenarios:

- If the file does not exist when the job starts, the job fails.
- If the file exists when the job starts and is then deleted without modification, the job fails.

Specifying the owner or group in the job definition:

If the file owner or group is specified in the job definition, the file trigger occurs only when all specified criteria are satisfied, including the owner or group criteria. If the owner or group criteria does not match, the job fails.

Monitoring a file continuously:

If continuous monitoring (continuous: Y) is specified and a wildcard is used in the file name value, file selection occurs after each trigger. The first matching file with a modification time later than the last trigger time is monitored.

### Example: Monitor for an Update to a File in a Directory and its Subdirectories

This example monitors the /usr/data/ directory and its subdirectories for the payroll file. When the payroll file is updated in any of the monitored directories, the job completes.

```
insert_job: ft_unix_update
job_type: FT
machine: unixagt
watch_file: /usr/data/payroll
watch_file_type: UPDATE
watch_file_recursive: Y
```

### Example: Monitor for Updates to All Files in a Directory and its Subdirectories

This example monitors all the files in the /usr/data/ directory and its subdirectories. When any file is updated in any of the monitored directories, the job completes.

```
insert_job: ft_unix_update
job_type: FT
machine: unixagt
watch_file: "/usr/data/*"
watch_file_type: UPDATE
watch_file_recursive: Y
```

## GENERATE File Trigger Type Notes

When specifying the GENERATE value in the watch\_file\_type attribute, consider the following points:

- If the file exists when the job starts, the trigger occurs when the file stops changing.
- If the file does not exist when the job starts, the trigger occurs when the file is created and stops changing.
- The file must remain unchanged for the amount of time specified in the watch\_no\_change attribute.
- If the specified directory does not exist or is deleted during monitoring, the job fails.
- If the file owner or group is specified in the job definition, the file trigger occurs only when all specified criteria are satisfied, including the owner or group criteria. If the owner or group criteria does not match, the job fails.
- If the file name specified in the job definition contains wildcards, the first matching file is selected.
- If continuous monitoring (continuous: Y) is specified and a wildcard is used in the file name value, each file that matches the name is monitored. An alert is issued for any file that is inactive for the specified watch\_no\_change period.

### Example: Monitor a File for Inactivity

This example monitors the file named companyXYZ.dat.

```
insert_job: FTP_input_companyXYZ
job_type: FT
machine: localhost
watch_file: /in_coming/FTP/companyXYZ.dat
watch_file_type: GENERATE
watch_no_change: 3
```

The job completes as follows:

- If the file exists, the job completes when the file remains unchanged for 3 minutes.
- If the file does not exist when the job starts, the job completes when the file is created and remains unchanged for 3 minutes.

## watch\_file\_win\_user Attribute—Specify a Windows User for Monitoring UNC Files

The `watch_file_win_user` attribute specifies the Windows user ID that monitors a network resource. For example, you can specify UNC names and share names in your job definition. The Windows user ID is used to monitor the specified file.

Usually, when running a Windows program as a service, you are restricted to how you can access data on remote computers. For example, to access data on a remote computer as a specified user ID, you must run the Windows service with that user ID. With the agent, however, those restrictions do not apply. Instead of running the agent service with a specific user ID, you can specify the user ID in the job definition.

**Note:** To use the `watch_file_win_user` attribute, you *cannot* specify the `watch_file_owner` and `watch_file_groupname` attributes in the same job definition.

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217) on Windows.

### Syntax

This attribute has the following format:

```
watch_file_win_user: user
user
```

Specifies the Windows user ID used to monitor the remote file.*user* ID and the domain the user ID belongs to.

**Limits:** Up to 80 characters; case-sensitive; it cannot contain delimiters (such as spaces)

### Example: Monitor a Remote Windows File

In this example, the path `c:\WINNT\Profiles\Visitor\Desktop\` has the share name `MyDesktop`. The command `notify.txt` is in that path on the CYBNT server. `JDOE` is a user ID in the CYBDOM domain and has access to the `notify.txt` command. `JDOE`'s password is defined on the scheduling manager. The job uses `JDOE` to monitor for the existence of the `notify.txt` file. If the file exists in the directory, the job completes. If the file does not exist in that directory, the job fails.

```
insert_job: ft_job
job_type: FT
machine: winagent
watch_file: \\CYBNT\MyDesktop\notify.txt
watch_file_type: EXIST
watch_file_win_user: CYBDOM\JDOE
```

## watch\_interval Attribute—Specify the Frequency to Monitor a File

The `watch_interval` attribute specifies how often the job checks the existence and size of the watched file.

### Supported Job Type

This attribute is optional for the [File Watcher \(FW\) job type](#) (see page 219).

### Syntax

This attribute has the following format:

`watch_interval: seconds`

#### **seconds**

Specifies the frequency (in seconds) the File Watcher job checks for the existence and size of the watched file.

**Default:** 60

**Limits:** Up to 2,147,483,647

### Notes:

- If you are monitoring for the existence of a file (not the size) and the file already exists when the job runs, the job completes immediately. The `watch_interval` attribute is ignored.
- The job is considered complete when the watched file reaches the minimum size and remains in a steady state (has not grown) during the specified interval. We recommend that you specify a reasonable interval to help ensure that the file does not appear to be in a steady state when it is still being modified over a longer period of time.
- If the FW job runs on the machine where CA WA Agent for UNIX, Linux, Windows, or i5/OS is installed, the agent uses the `watch_interval` as a second polling interval. The first polling interval is set by default on the agent for 30 seconds. If the job does not complete after the first polling interval, the agent waits for the second polling interval specified by the `watch_interval` attribute to ensure the file remains steady. If the file has not changed after the second polling interval elapses, the agent returns the status. If the file does change, the agent goes back to sleep for the duration of the second polling interval until the file eventually stabilizes.

#### Example: Monitor a File Every 120 Seconds

This example monitors the watch\_file.log file on the winagent computer. The job completes when the file reaches 10,000 bytes and maintains a steady state for at least 120 seconds (30 seconds for the agent's global poll interval plus 90 seconds for the watch\_interval).

```
insert_job: fw_job
job_type: FW
machine: winagent
watch_file: "c:\tmp\watch_file.log"
watch_file_min_size: 10000
watch_interval: 90
```

#### Monitor a File Every 60 Seconds on a Legacy Agent

This example monitors the watch\_file.log file on the unixagent computer. The unixagent is a legacy agent. The job completes when the file reaches 10000 bytes and maintains a steady state for 60 seconds.

```
insert_job: fw_job
job_type: FW
machine: unixagent
watch_file: /tmp/watch_file.log
watch_file_min_size: 10000
watch_interval: 60
```

## watch\_no\_change Attribute—Specify the Time the File Must Remain Unchanged

The watch\_no\_change attribute defines the time the file must remain unchanged to satisfy the monitor condition.

### Supported Job Type

This attribute is optional for the [File Trigger \(FT\) job type](#) (see page 217).

**Note:** This attribute is only valid when CREATE, EXPAND, SHRINK, UPDATE, or GENERATE is specified for the watch\_file\_type attribute.

### Syntax

This attribute has the following format:

watch\_no\_change: *interval*

#### *interval*

Defines the number of minutes the file must remain unchanged to satisfy the monitor condition.

**Limits:** Up to 2,147,483,647

**Default:** 1

### Example: Monitor for an Increase in File Size to a Certain Amount with an Unchanged Condition

In this example, the unixagt agent monitors the analysis file in the /research directory. If the file size expands to 1 byte or more and remains unchanged for 120 minutes or more, the job completes.

```
insert_job: ft_unix_nochange
job_type: FT
machine: unixagt
watch_file: /research/analysis
watch_file_type: EXPAND
watch_file_change_type: SIZE
watch_file_change_value: 1
watch_no_change: 120
```

## web\_parameter Attribute—Specify Operation Parameters

The web\_parameter attribute specifies the operation parameters in a Web Service job. When the job invokes the web service, the parameters are passed to the operation.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

```
web_parameter: xsd\{:type=value | payload_job=job_name[, xsd\{:type=value |
payload_job=job_name...]
```

**xsd\:*type*=*value***

Specifies the XML schema type and value for the operation parameter. The agent supports the following types:

- xsd\:string
- xsd\:integer
- xsd\:int
- xsd\:long
- xsd\:short
- xsd\:decimal
- xsd\:float
- xsd\:double
- xsd\:boolean
- xsd\:byte
- xsd\:unsignedInt
- xsd\:unsignedShort
- xsd\:unsignedByte
- xsd\:QName
- xsd\:dateTime
- xsd\:date
- xsd\:time
- xsd\:anyURI
- xsd\:base64Binary
- xsd\:hexBinary
- xsd\:anySimpleType
- xsd\:duration
- xsd\:gYearMonth
- xsd\:gYear
- xsd\:gMonthDay
- xsd\:gDay
- xsd\:gMonth
- xsd\:normalizedString
- xsd\:token
- xsd\:language
- xsd\:Name
- xsd\:NCName
- xsd\:ID
- xsd\:NMTOKEN
- xsd\:NMTOKENS
- xsd\:nonPositiveInteger
- xsd\:negativeInteger
- xsd\:nonNegativeInteger
- xsd\:unsignedLong
- xsd\:positiveInteger

**Limits:** Up to 1024 characters; case-sensitive

**payload\_job=*job\_name***

Specifies the name of the payload producing job that produced the binary output to be used as an input parameter. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object can be passed as input to a payload consuming job.

**Notes:**

- You can define up to 64 parameters using one web\_parameter attribute or multiple entries of web\_parameter.
- In each web\_parameter entry, separate each set of parameter values with a comma. The entire value of each entry can be up to 4096 characters.
- Order is important. The agent passes the parameters in the order that you specify them.

**Example: Pass Parameter to Web Service for Getting Stock Quotes**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is <http://www.webservicex.com/stockquote.asmx?WSDL>. The WSDL port name within the target namespace <http://www.webservicex.NET> is StockQuoteSoap. The target endpoint address URL is <http://www.webservicex.com/stockquote.asmx>. The job calls the operation GetQuote within the StockQuote web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The GetQuote operation returns a java.lang.String object, which maps to the XML type string in the return namespace <http://www.webservicex.NET/>. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webservicex.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webservicex.NET/"
```

## web\_user Attribute—Specify a Web Service User Name

The web\_user attribute specifies the user name in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

`web_user: user_name`

***user\_name***

Specifies the user name.

**Limits:** Up to 145 characters

## win\_event\_category Attribute—Specify a Windows Event Category

The `win_event_category` attribute specifies the event category as displayed in the Windows Event Viewer.

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

`win_event_category: event_category`

#### *event\_category*

Specifies the event category as displayed in the Windows Event Viewer.

**Limits:** Up to 256 characters; case-sensitive

### Example: Monitor for a Windows Application Event Log

This example monitors for a Windows application event log. When the job finds an application log in the DISK category and has an event ID equal to 14, the job completes.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_category: DISK
win_event_op: EQ
win_event_id: 14
```

## win\_event\_computer Attribute—Specify a Local Computer for Windows Event Log

The `win_event_computer` attribute specifies the name of the local machine that the Windows event log applies to.

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

`win_event_computer: computer_name`

**`computer_name`**

Specifies the local computer name where the event log is monitored.

**Limits:** Up to 80 characters; case-sensitive

### Example: Monitor a Windows Event Log on a Specific Computer

This example monitors the D9GBJG11 computer for an event source of Security.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Security
win_event_type: AUDITS
win_event_source: Security
win_event_computer: D9GBJG11
win_event_category: System Event
win_event_op: GT
win_event_id: 0
```

## win\_event\_datetime Attribute—Specify the Date and Time of a Windows Event Log

The `win_event_datetime` attribute specifies the date and time of a Windows event log. The job monitors for an event log that occurs on or after the specified date and time.

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

`win_event_datetime: "yyyymmdd hh:mm:ss"`

**"yyyymmdd hh:mm:ss"**

Specifies the date and time of the Windows event log. This parameter includes the following:

**yyyy**

Specifies the four-digit year.

**mm**

Specifies the two-digit month.

**dd**

Specifies the two-digit day.

**hh**

Specifies the two-digit hour.

**mm**

Specifies the two-digit minute.

**ss**

Specifies the two-digit second.

Example: **"20090501 06:30:00"**

#### **Example: Monitor an Application Log that Occurs on or After a Specified Date**

This example monitors an application log that occurs any time on or after July 1, 2010. When this event occurs in the application log, the job completes successfully.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_datetime: "20100701 00:00:00"
```

## **win\_event\_description Attribute—Specify a Windows Event Description**

The `win_event_description` attribute specifies the description of the Windows event as shown in the Event Viewer.

### **Supported Job Type**

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### **Syntax**

This attribute has the following format:

```
win_event_description: event_description
event_description
```

Specifies the event description as displayed in the Windows Event Viewer.

**Limits:** Up to 256 characters; case-sensitive

#### **Example: Monitor for Words Excluded in the Event Description**

In this example, the event description must include the words conflict and state as indicated by the plus signs but must exclude the words deny, master, or browser as indicated by the minus sign. The plus sign is the default and is optional.

```
insert_job: eventlog_job1
job_type: OMEL
machine: monagt
win_log_name: System
win_event_type: ERROR
win_event_description: "+conflict +state -deny -master -browser"
```

#### Example: Monitor for Words Included in the Event Description

In this example, the event description must include the words Normal shutdown. This example does not use the plus sign, and by default, the specified words are included in the search.

```
insert_job: eventlog_job2
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_type: INFO
win_event_description: "Normal shutdown"
```

## win\_event\_id Attribute—Specify a Windows Event ID

The `win_event_id` attribute specifies the Windows event ID.

#### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

#### Syntax

This attribute has the following format:

`win_event_id: id`

***id***

Specifies the Windows event ID to monitor.

**Limits:** Up to 10 digits

**Example:** 4000

#### Example: Monitor for an Event ID Less Than a Specified Value

This example checks for an event ID number less than 1. The job returns the first application event from the application log that has an event ID equal to 0.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_op: LT
win_event_id: 1
```

## win\_event\_op Attribute—Specify a Comparison Operator for a Windows Event ID

The `win_event_op` attribute specifies a comparison operator against the value of a Windows Event ID.

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

`win_event_op: GT | LT | LE | EQ | GE`

#### **EQ**

Specifies the equal to (=) operator against the value of a Windows Event ID. This is the default.

#### **LE**

Specifies the less than or equal to (<=) operator against the value of a Windows Event ID.

#### **LT**

Specifies the less than (<) operator against the value of a Windows Event ID.

#### **GE**

Specifies the greater than or equal to (>=) operator against the value of a Windows Event ID.

#### **GT**

Specifies the greater than (>) operator against the value of a Windows Event ID.

### Example: Monitor for an Event ID Less Than a Specified Value

This example checks for an event ID number less than 1. The job returns the first application event from the application log that has an event ID equal to 0.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_op: LT
win_event_id: 1
```

## win\_event\_source Attribute—Specify a Windows Event Source

The win\_event\_source attribute specifies the Windows event source.

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

```
win_event_source: source_name
source_name
```

Specifies the event source as displayed in the Windows Event Viewer.

**Limits:** Up to 256 characters; case-sensitive

### Example: Monitor a Windows Event Source

This example monitors a Windows event source named Service Control Manager.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: System
win_event_type: ERROR
win_event_source: Service Control Manager
```

## win\_event\_type Attribute—Specify a Windows Event Type

The `win_event_type` attribute specifies the event type to monitor in the Windows event log.

**Note:** If you do not specify the `win_event_type` attribute in your job definition, the job monitors for the Error event type (the default).

### Supported Job Type

This attribute is optional for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

### Syntax

This attribute has the following format:

`win_event_type: ERROR | WARN | INFO | AUDITS | AUDITF`

#### **ERROR**

Specifies the Error event type. This is the default.

#### **WARN**

Specifies the Warning event type.

#### **INFO**

Specifies the Information event type.

#### **AUDITS**

Specifies the Success Audit event type.

#### **AUDITF**

Specifies the Failure Audit event type.

### Example: Monitor an Application Event Log for Info Events

This example monitors the Application log for INFO events.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Application
win_event_type: INFO
win_event_category: None
win_event_source: Userenv
win_event_computer: LLUSER
win_event_id: 1000
win_event_op: EQ
```

#### Example: Monitor for a Successful Audit

In this example, the security log is monitored for a successful audit of a security access attempt. The event category is System Event, the term succeeded is excluded, but the words Audit and log are included in the event description.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: Security
win_event_type: AUDITS
win_event_category: System Event
win_event_source: Service Control Manager
win_event_description: "-succeeded +Audit log"
```

## win\_log\_name Attribute—Specify the Name of a Windows Event Log

The win\_log\_name attribute specifies the name of a Windows event log to monitor.

#### Supported Job Type

This attribute is required for the [Windows Event Log Monitoring \(OMEL\) job type](#) (see page 279).

#### Syntax

This attribute has the following format:

win\_log\_name: *log\_name*

#### *log\_name*

Specifies the name of the event log.

**Limits:** Up to 256 characters; case-sensitive

#### Example: Monitor a Windows Event Log named System

This example monitors a system event log for an event type of WARN, event source of MrxSmb, and event category of None.

```
insert_job: eventlog_job
job_type: OMEL
machine: monagt
win_log_name: System
win_event_type: WARN
win_event_source: MrxSmb
win_event_category: None
```

## win\_service\_name Attribute—Specify the Name of the Windows Service to be Monitored

The `win_service_name` attribute specifies the name of the Windows service to be monitored.

### Supported Job Type

This attribute is required for the [Windows Service Monitoring \(OMS\) job type](#) (see page 281).

### Syntax

This attribute has the following format:

`win_service_name: service_name`

#### *service\_name*

Specifies the name of the local Windows service to be monitored.

**Limits:** Up to 256 characters; case-sensitive

**Note:** You can only specify the service name. The service display name is shown in the Windows Services Manager.

### Example: Specify a Windows Service Name

This example monitors a Windows service named Proc Server. The job checks the status immediately and completes successfully if the service is paused. If the service is not paused, the job fails.

```
insert_job: oms_job1
job_type: OMS
machine: winagt
win_service_name: Proc Server
win_service_status: PAUSED
monitor_mode: NOW
```

### Example: Specify a Path to a Windows Service Executable

This example monitors a Windows service named Log App. The job waits until the service status is CONTINUE\_PENDING before it completes.

```
insert_job: oms_job2
job_type: OMS
machine: winagt
win_service_name: "C:\Program Files\Log App\apptask.exe"
win_service_status: CONTINUE_PENDING
monitor_mode: WAIT
```

## win\_service\_status Attribute—Specify the Status of the Windows Service to be Monitored

The `win_service_status` attribute specifies the status of the Windows Service to be monitored. The job checks if the status of the service matches the status specified by this attribute.

**Note:** If you do not specify the `win_service_status` attribute in your job definition, the job checks if the Windows service is running (the default).

### Supported Job Type

This attribute is optional for the [Windows Service Monitoring \(OMS\) job type](#) (see page 281).

### Syntax

This attribute has the following format:

```
win_service_status: RUNNING | STOPPED | CONTINUE_PENDING |
 PAUSE_PENDING | PAUSED | START_PENDING |
 STOP_PENDING | EXISTS | NOTEXISTS
```

#### RUNNING

Specifies that the job monitors the Windows service for a running status. This is the default.

#### **STOPPED**

Specifies that the job monitors the Windows service for a stopped status.

#### **CONTINUE\_PENDING**

Specifies that the job monitors the Windows service for a continue pending status.

#### **PAUSE\_PENDING**

Specifies that the job monitors the Windows service for a pause pending status.

#### **PAUSED**

Specifies that the job monitors the Windows service for a paused status.

#### **START\_PENDING**

Specifies that the job monitors the Windows service for a start pending status.

#### **STOP\_PENDING**

Specifies that the job monitors the Windows service for a stop pending status.

#### **EXISTS**

Specifies that the job checks whether the Windows service exists.

#### NOTEXISTS

Specifies that the job checks whether the Windows service does not exist.

#### **Example: Monitor for the Existence of a Windows Service**

This example monitors a Windows service named Proc Server. The job completes successfully if the service exists. By default, the job checks for the condition immediately and completes, so the monitor\_mode attribute is not required in the job definition.

```
insert_job: oms_job1
job_type: OMS
machine: winagt
win_service_name: Proc Server
win_service_status: EXISTS
```

#### **Example: Monitor for a Running Status**

This example monitors a Windows service named App Server. The win\_service\_status attribute is not specified in the job definition, so the job monitors for a RUNNING status by default. The job completes when the service is running.

```
insert_job: oms_job2
job_type: OMS
machine: winagt
win_service_name: App Server
monitor_mode: WAIT
```

## **wsdl\_operation Attribute—Specify the Operation to be Invoked**

The wsdl\_operation attribute specifies the web service operation to be invoked in a Web Service job.

#### **Supported Job Type**

This attribute is required for the [Web Service \(WBSVC\) job type](#) (see page 276).

#### **Syntax**

This attribute has the following format:

`wsdl_operation: operation`

#### ***operation***

Specifies the operation to be invoked.

**Limits:** Up to 256 characters; case-sensitive

### Example: Invoke a Web Service Operation

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webservicex.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type `string` in the return namespace `http://www.webservicex.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## WSDL\_URL Attribute—Specify a WSDL URL

The WSDL\_URL attribute specifies the URL to the Web Service Description Language (WSDL) of the web service to invoke in a Web Service job.

### Supported Job Type

This attribute is optional for the [Web Service \(WBSVC\) job type](#) (see page 276).

### Syntax

This attribute has the following format:

WSDL\_URL: *wsdl\_url*

***wsdl\_url***

Specifies the URL to the Web Service Description Language (WSDL) of the web service to invoke.

**Limits:** Up to 256 characters; case-sensitive

**Note:** In a Web Service job, if you specify the WSDL\_URL attribute but not the endpoint\_URL attribute, you must specify both the service\_name and port\_name attributes. For the job to run successfully without the endpoint\_URL attribute, the agent must be running on the same computer as the application server such as WebLogic or JBoss. If you specify both the WSDL\_URL and endpoint\_URL attributes, then the service\_name and port\_name attributes are optional.

**Example: Specify WSDL URL for Getting Stock Quotes**

Suppose that you want to invoke a web service that returns a company stock quote. The URL for the WSDL that describes the web service and its location is `http://www.webservicex.com/stockquote.asmx?WSDL`. The WSDL port name within the target namespace `http://www.webserviceX.NET` is `StockQuoteSoap`. The target endpoint address URL is `http://www.webservicex.com/stockquote.asmx`. The job calls the operation `GetQuote` within the `StockQuote` web service. When the job invokes the web service, the company's stock symbol is passed to the operation. The `GetQuote` operation returns a `java.lang.String` object, which maps to the XML type `string` in the return namespace `http://www.webserviceX.NET/`. When the job completes, the stock quote for CA is stored as a serialized Java object in the job's spool directory.

```
insert_job: quote
job_type: WBSVC
machine: wsagent
target_namespace: "http://www.webserviceX.NET/"
service_name: StockQuote
port_name: StockQuoteSoap
wsdl_operation: GetQuote
one_way: FALSE
WSDL_URL: "http://www.webservicex.com/stockquote.asmx?WSDL"
endpoint_URL: "http://www.webservicex.com/stockquote.asmx"
web_parameter: xsd\:string="CA"
return_class_name: java.lang.String
return_xml_name: string
return_namespace: "http://www.webserviceX.NET/"
```

## zos\_dataset Attribute—Specify a JCL Library for a z/OS Data Set Trigger Job

The `zos_dataset` attribute specifies the JCL library that contains the JCL for your job.

### Supported Job Type

This attribute is required for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

### Syntax

This attribute has the following format:

`zos_dataset: jcl_library`

#### *jcl\_library*

Specifies the Job Control Language (JCL) library name. The JCL library or JCLLIB contains the JCL for the z/OS job. The JCLLIB is a z/OS data set name. Data set names typically have up to four positions; positions must each be eight characters or less and separated by periods.

**Limits:** Up to 44 characters

**Example:** `prod.jcllib`

**Note:** Instead of specifying a single data set, you can also use the hyphen wildcard to specify a group of data sets that match the selection criteria. For example, the following table shows how the hyphen wildcard matches a group of data sets:

| Specified data set name | Matching data sets                                                                                   |
|-------------------------|------------------------------------------------------------------------------------------------------|
| CYB1.PAYROLL.A          | CYB1.PAYROLL.A                                                                                       |
| CYB1.PAYROLL.G-         | CYB1.PAYROLL.G0145V00<br>CYB1.PAYROLL.G0146V00<br>CYB1.PAYROLL.G0147V00<br>CYB1.PAYROLL.GRANTED.MORE |
| CYB1.P-                 | CYB1.PAYROLL<br>CYB1.POSTJOBS<br>CYB1.PROD.INPUT                                                     |

#### Example: Specify a JCL Library

Suppose that you want a z/OS Data Set Trigger job named PROD.NIGHTLY to release its successors when the data set PROD.CICS.FILE1602 is closed (created or updated). The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.NIGHTLY
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: PROD.CICS.FILE1602
```

## zos\_dsn\_renamed Attribute—Specify Whether to Monitor When a Data Set is Renamed

The zos\_dsn\_renamed attribute specifies whether the job monitors when a data set is renamed. When the data set is renamed, the job triggers.

**Note:** If you specify the zos\_dsn\_renamed attribute in your job definition, you cannot specify the zos\_dsn\_updated attribute.

#### Supported Job Type

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

#### Syntax

This attribute has the following format:

zos\_dsn\_renamed: Y | N

**Y**

Monitors when the specified data set is renamed.

**N**

Does not monitor when the specified data set is renamed. This is the default.

#### **Example: Monitor When a Data Set is Renamed**

This example monitors a data set named CYB2.QA.DSTRIG1. When the data set is renamed, the job triggers.

```
insert_job: test_ZOSDST
job_type: ZOSDST
machine: zosagent
owner: user1@ca11
zos_dsn_updated: N
zos_trigger_on: 0
zos_dataset: CYB2.DA.DSTRIG1
zos_dsn_renamed: y
```

## **zos\_dsn\_updated Attribute—Specify Whether to Monitor for Updates to a Data Set**

The `zos_dsn_updated` attribute specifies whether the job monitors for updates to a data set. When the data set is updated, the job triggers.

**Note:** If you specify the `zos_dsn_updated` attribute in your job definition, you cannot specify the `zos_dsn_renamed` attribute.

#### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

#### **Syntax**

This attribute has the following format:

`zos_dsn_updated: Y | N`

**Y**

Monitors for updates to the specified data set.

**N**

Does not monitor for updates to the specified data set. This is the default.

#### **Example: Monitor When a Data Set is Updated**

This example monitors a data set named CYB2.QA.DSTRIG1. When the data set is updated, the job triggers.

```
insert_job: test_ZOSDST_Upd
job_type: ZOSDST
machine: zosagent
owner: user1@ca11
zos_dsn_updated: Y
zos_dataset: CYB2.DA.DSTRIG1
```

## **zos\_explicit\_dsn Attribute—Specify Whether to Monitor for an Explicit Data Set Notification**

The `zos_explicit_dsn` attribute specifies whether the job monitors for an explicit data set notification (used when the data set activity does not generate an SMF record). The job triggers when it receives notification that the specified data set is being updated.

**Note:** If you do not specify the `zos_explicit_dsn` attribute in your job definition, the job does not monitor for an explicit data set notification (the default).

#### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

#### **Syntax**

This attribute has the following format:

`zos_explicit_dsn: TRUE | FALSE`

##### **TRUE**

Monitors for an explicit data set notification.

##### **FALSE**

Does not monitor for an explicit data set notification. This is the default.

#### **Example: Monitor for an Explicit Data Set Notification**

This example monitors a data set named CYB2.DA.DSTRIG1. When the job receives notification that the data set is being updated, the job completes.

```
insert_job: test_ZOSDST
job_type: ZOSDST
machine: zosagent
owner: user1@ca11
zos_dsn_updated: N
zos_trigger_on: 0
zos_dataset: CYB2.DA.DSTRIG1
zos_explicit_dsn: y
```

## **zos\_ftp\_direction Attribute—Specify Whether to Monitor for an FTP Transfer To or From a Remote Computer**

The `zos_ftp_direction` attribute specifies whether the job monitors for an FTP transfer to a remote computer or from a remote computer.

**Note:** To use this attribute, you must specify FALSE for the `zos_explicit_dsn` attribute.

#### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

#### **Syntax**

This attribute has the following format:

```
zos_ftp_direction: RECEIVE | SEND
```

##### **RECEIVE**

Monitors for an FTP transfer from a remote computer to the local mainframe computer.

##### **SEND**

Monitors for an FTP transfer from the local mainframe computer to a remote computer.

**Example: Monitor for a Data Set Received from a Remote FTP Partner**

Suppose that you want the z/OS Data Set Trigger job PROD.PAY\_DATA to release its successors when a file is successfully received from a remote FTP partner, creating generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the FTP transfer under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: USER1.PAYROLL.G-
zos_ftp_direction: RECEIVE
```

## zos\_ftp\_host Attribute—Specify a Remote Computer for an FTP Transfer

The zos\_ftp\_host attribute specifies the remote computer used in an FTP transfer. The job only triggers when the FTP transfer occurs on the specified computer. The job does not trigger for FTP transfers on other computers.

**Note:** To use this attribute, you must specify FALSE for the zos\_explicit\_dsn attribute.

### Supported Job Type

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

### Syntax

This attribute has the following format:

zos\_ftp\_host: *host\_name*

***host\_name***

Specifies the name of the remote computer involved in the FTP transfer. The data is transferred to or from the local mainframe computer.

**Limits:** Up to 128 characters

#### **Example: Restrict Triggering to a Specific Host**

Suppose that you want the z/OS Data Set Trigger job CYBER.XFER to release its successors when a remote FTP partner with IP address 172.16.0.0 successfully transfers a file creating the data set CYBER.XFER.001. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

```
insert_job: CYBER.XFER
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: CYBER.XFER.001
zos_ftp_direction: RECEIVE
zos_ftp_host: 172.16.0.0
zos_ftp_userid: CYB1
```

## **zos\_ftp\_userid Attribute—Specify a User ID for an FTP Connection**

The `zos_ftp_userid` attribute specifies an FTP user ID used to connect to a remote computer. The job only triggers for FTP transfers by the specified logon user ID. The job does not trigger for FTP transfers by other user IDs.

**Note:** To use this attribute, you must specify FALSE for the `zos_explicit_dsn` attribute.

#### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

#### **Syntax**

This attribute has the following format:

`zos_ftp_userid: user`

*user*

Specifies the FTP user ID used to connect to a remote computer.

**Limits:** Up to 80 characters

### Example: Restrict Triggering to a Specific Logon ID

Suppose that you want the z/OS Data Set Trigger job CYBER.XFER to release its successors when a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming that the remote FTP partner logged on to the FTP server with the CYBER005 user ID. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

```
insert_job: CYBER.XFER
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: CYBER.XFER.001
zos_ftp_direction: RECEIVE
zos_ftp_host: 172.16.0.0
zos_ftp_userid: CYBER005
```

## zos\_jobname Attribute—Specify the z/OS Job Name

The zos\_jobname attribute specifies the name of the z/OS job that is being monitored. The job is submitted outside of CA Workload Automation AE, such as a job that is submitted manually by a user.

### Supported Job Type

This attribute is required for the [z/OS Manual \(ZOSM\) job type](#) (see page 284).

### Syntax

This attribute has the following format:

`zos_jobname: name`

#### *name*

Specifies the name of the z/OS job that is being monitored.

**Limits:** Up to 8 alphanumeric or national (\$, #, @) characters. The first character must be alphabetic or national. If your z/OS system uses ANSI tapes, the name must contain only alphanumeric characters.

**Example: Post a z/OS Manual Job as Complete Based on the User ID**

This example posts a z/OS Manual job as complete when the manually-submitted job ABC runs under user CYBER. The ZOS1 agent monitors job ABC.

```
insert_job: ABC_job
job_type: ZOSM
machine: ZOS1
zos_jobname: ABC
owner: zosuser
auth_string: CYBER
```

## **zos\_trigger\_by Attribute—Specify a Job Name or a User ID that Triggers the Job**

The `zos_trigger_by` attribute specifies the name of the job or the user who performs the data set activity that triggers the job. The job trigger is restricted to data set activity performed by the specified job or user.

**Note:** To use this attribute, you must also specify the `zos_trigger_type` attribute in your job definition.

### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

### **Syntax**

This attribute has the following format:

```
zos_trigger_by: name
name
```

Specifies the name of the job or user who performs the data set activity that triggers the job.

**Limits:** Up to 128 characters

**Example: Restrict the Trigger to Specific Data Sets Created by a Particular Job**

Suppose that you want a z/OS Data Set Trigger job named PROD.PAY\_DATA to release its successors when job ABC creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: USER1.PAYROLL.G-
zos_trigger_type: zos_job_name
zos_trigger_by: ABC
```

**Example: Restrict the Trigger to Specific Data Sets Created by a Particular User**

Suppose that you want the z/OS Data Set Trigger job PROD.PAY\_DATA to release its successors when the user CYB1 creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: USER1.PAYROLL.G-
zos_trigger_type: zos_user_id
zos_trigger_by: CYB1
```

**Example: Restrict the Trigger to an FTP Transfer from a Specific User ID**

This example releases the job's successors when a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming that the user ID prefix of the local FTP partner is CYB (CYB-). The agent ZOS1 monitors the FTP transfer under user CYBDL01.

```
insert_job: CYBER.XFER
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: CYBER.XFER.001
zos_trigger_type: zos_user_id
zos_trigger_by: CYB-
zos_ftp_direction: RECEIVE
zos_ftp_userid: CYB-
```

## zos\_trigger\_on Attribute—Specify the Number of Actions that Must Occur

The zos\_trigger\_on attribute specifies the number of actions that must occur before the job triggers.

### Supported Job Type

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

### Syntax

This attribute has the following format:

`zos_trigger_on: number`

***number***

Specifies the number of actions that must occur before the job triggers.

**Limits:** 0-100

### Example: Run a Compress Job After 100 Closures of a Data Set

Suppose that you want the z/OS Data Set Trigger job PROD.PAY\_DATA to release a compress job named COPYJCL.COMPRESS after every 100 closures of the data set CYBER.COPY.JCL. The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: CYBER.COPY.JCL
zos_trigger_on: 100
```

**Note:** Define the compress job, COPYJCL.COMPRESS, as a successor to the z/OS Data Set Trigger job.

## **zos\_trigger\_type Attribute—Specify Whether to Monitor for Data Set Activity by a Job or a User ID**

The zos\_trigger\_type attribute specifies whether the job monitors data set activity by a job or a user ID. When a data set is updated or renamed by the specified job or user ID, the job triggers.

**Note:** To use this attribute, you must also specify the zos\_trigger\_by attribute in your job definition.

### **Supported Job Type**

This attribute is optional for the [z/OS Data Set Trigger \(ZOSDST\) job type](#) (see page 282).

### **Syntax**

This attribute has the following format:

**zos\_trigger\_type: zos\_job\_name | zos\_user\_id**

#### **zos\_job\_name**

Monitors data set activity by a job.

#### **zos\_user\_id**

Monitors data set activity by a user ID.

### **Example: Restrict the Trigger to Specific Data Sets Created by a Particular Job**

Suppose that you want a z/OS Data Set Trigger job named PROD.PAY\_DATA to release its successors when job ABC creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-).The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: USER1.PAYROLL.G-
zos_trigger_type: zos_job_name
zos_trigger_by: ABC
```

**Example: Restrict the Trigger to Specific Data Sets Created by a Particular User**

Suppose that you want the z/OS Data Set Trigger job PROD.PAY\_DATA to release its successors when the user CYB1 creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set under user CYBDL01.

```
insert_job: PROD.PAY_DATA
job_type: ZOSDST
machine: ZOS1
owner: CYBDL01
zos_dataset: USER1.PAYROLL.G-
zos_trigger_type: zos_user_id
zos_trigger_by: CYB1
```

# Chapter 5: JIL User-Defined Job Type Definitions

---

This chapter describes the JIL subcommands and attributes that you can use to define and update, and delete CA Workload Automation AE job types. To define, update, or delete a job type, you specify the appropriate JIL subcommand and attribute statements to the `jil` command.

## delete\_job\_type Subcommand—Delete a Job Type from the Database

The `delete_job_type` subcommand deletes a job type from the database. Before deleting the job type, make sure no job is currently defined with this type.

### Syntax

This subcommand has the following format:

```
delete_job_type: value
value
```

Defines the value between 0-9, of the job type that is currently defined in the database.

### Example: Delete a Job Type from the Database

This example deletes the job type 1:

```
delete_job_type: 1
```

## insert\_job\_type Subcommand—Add a Job Type to the Database

The insert\_job\_type subcommand adds a new job type definition to the database. The job type creates an association to an executable.

The command attribute value is required, whereas the description attribute is optional. If the job also contains a command attribute, it will be concatenated to the insert\_job\_type subcommand.

**Note:** A job type must be defined before it can be used in a job definition.

### Syntax

This subcommand has the following format:

```
insert_job_type: value
value
```

Defines a job type.

**Limits:** 0-9

### Example: Add a Job Type with a Command to the Database

This example creates the job type with a binary file wv:

```
insert_job_type: 1
command: /tmp/wv
```

### More information:

[command Attribute—Specify Command for the Job Type](#) (see page 807)

## update\_job\_type Subcommand—Update a Job Type in the Database

The update\_job\_type subcommand alters a user-defined job type definition in the database. The command attribute value is required, whereas the description attribute is optional.

### Syntax

This subcommand has the following format:

```
update_job_type: value
value
```

Specifies the job type to update.

**Limits:** 0-9

### Example: Update a Job Type with a New Command of ftp to the Database

This example updates the job type with binary file ftp:

```
update_job_type: 1
command: /bin/ftp
```

### More information:

[command Attribute—Specify Command for the Job Type](#) (see page 807)

## command Attribute—Specify Command for the Job Type

The command attribute specifies the binary file to use for the job type.

### Syntax

This attribute has the following format:

```
command: file
file
```

Specifies the binary file to use for the command in a job.

**Limits:** Up to 255 characters

**Windows Note:** When specifying drive letters in job definitions, you must escape the colon with backslashes. For example, C:\\tmp is valid; C:\\tmp is not.

## **description Attribute—Specify a Description for the Job Type**

The `description` attribute specifies a description to use for the job type.

This attribute has the following format:

`description: text`

***text***

Specifies the text description for the job type.

**Limits:** Up to 255 characters

# Chapter 6: JIL Machine Definitions

---

This chapter describes the JIL subcommands and attributes that you can use to define and delete the machines where jobs run. You can define real and virtual machines in the Windows or UNIX operating environments. To define or delete a machine, you specify the appropriate JIL subcommand and attribute statements to the `jil` command.

## delete\_machine Subcommand—Delete a Real or Virtual Machine

The `delete_machine` subcommand deletes the following from the CA Workload Automation AE database:

- A real machine
- A virtual machine
- Virtual machine and real machine pool references to a real machine

### Syntax

This subcommand has the following format:

`delete_machine: machine_name`

***machine\_name***

Identifies a real or virtual machine that currently exists in the CA Workload Automation AE database.

### Optional Attributes

You can specify the following optional attributes when deleting a virtual machine:

- [machine](#) (see page 830)

You can specify the following optional attributes when deleting a real machine:

- [force](#) (see page 828)
- [remove\\_references](#) (see page 836)

**Note:** If CA Workload Automation AE is running in external security mode using CA EEM, you must have the following authority:

- DELETE authority for the corresponding as-machine resource policy to delete the real or virtual machine.
- WRITE authority for the corresponding as-machine resource policies to remove references to real machines or to update the virtual machines that reference it.

#### **Example: Delete a Machine**

This example deletes the machine mach1, which could be a virtual or a real machine.

```
delete_machine: mach1
```

#### **Example: Delete a Real Machine From the Virtual Machine**

This example deletes the real machine socrates from the virtual machine cheetah without deleting cheetah.

```
delete_machine: cheetah
machine: socrates
```

## **Delete a Real Machine**

To delete a real machine that is not referenced by any virtual machines, use the following syntax:

```
delete_machine: real_machine_name
```

To delete a real machine that is referenced by virtual machines, you must do *one* of the following:

- [Individually delete the references from all virtual machines that reference it](#) (see page 811).
- [Delete all references at one time and the real machine itself](#) (see page 812).

#### **Example: Delete a Real Machine**

This example deletes the real machine named realmach1. This machine is not referenced by any virtual machines.

```
delete_machine: realmach1
```

## Delete a Virtual Machine

To delete a virtual machine, use the following syntax:

```
delete_machine: virtual_machine_name
```

The virtual machine is deleted, but the real machines it references are not deleted.

## Delete a Real Machine Pool

To delete a real machine pool, use the following syntax:

```
delete_machine: real_machine_pool_name
```

The real machine pool is deleted, but the real machines it references are not deleted.

## Delete a Reference to a Real Machine

To delete a virtual machine or real machine pool reference to a real machine, use the following syntax:

```
delete_machine: virtual_machine_name
machine: real_machine_name_referenced
```

Or

```
delete_machine: real_machine_pool_name
machine: real_machine_name_referenced
```

The reference to the real machine is deleted. The real machine definition itself is *not* deleted.

**Note:** You can individually delete all real machine references in a virtual machine or a real machine pool until there is only one reference remaining. You cannot delete the last reference. If you try to delete the virtual machine's or real machine pool's last machine reference, the command will fail because a virtual machine or real machine pool definition with no references is invalid. To delete the last reference in a virtual machine or real machine pool, you must delete the virtual machine or real machine pool itself.

The real machine can be deleted after the previous command is issued for the last virtual machine or real machine pool that references it.

**Example: Delete a Real Machine that is the One of Several References of a Virtual Machine**

Suppose that a virtual machine named virtmach1 has references to multiple real machines. One of the references is to the real machine named realmach1. This example deletes realmach1.

```
delete_machine: virtmach1
machine: realmach1
```

## Delete All References to a Real Machine and the Machine Itself

Instead of individually deleting references to a real machine from virtual machines or real machine pools, you can delete all references at one time. The real machine itself is also deleted.

To delete the real machine and all references to it, use the following syntax:

```
delete_machine: real_machine_name
remove_references: y
```

**Note:** This command deletes all references to the specified real machine from all virtual machines or real machine pools even if a virtual machine or real machine pool only has that one reference. After that one reference is deleted, the virtual machine or real machine pool is considered empty. You cannot re-define that virtual machine or real machine pool because it already exists. However, you can re-insert real machine references to it. To display all empty virtual machines or real machine pools that you can use, issue the following command:

```
autorep -M ALL
```

**Example: Delete a Real Machine and All References to It**

Suppose that the real machine named realmach1 is referenced by virtual machines. This example deletes realmach1 and all references to it.

```
delete_machine: realmach1
remove_references: y
```

## insert\_machine Subcommand—Add a Machine Definition

The insert\_machine subcommand adds a new machine definition to the CA Workload Automation AE database for a real machine or virtual machine.

You can also add a machine definition for the following agents:

- CA Workload Automation Agent for UNIX, Linux, Windows, i5/OS, or z/OS
- CA NSM Job Management Option
- CA Universal Job Management Agent (UUJMA)
- CA AutoSys Workload Automation Connect Option

After you define a machine, you can schedule jobs to run on those machines. You can identify any machine accessible through the TCP/IP protocol in the machine attribute of a job.

### Notes:

- You must define the machine using the insert\_machine subcommand before you can insert a job that uses that machine. The job will not insert if the machine definition does not exist.
- If CA Workload Automation AE is running in external security mode using CA EEM, you must have CREATE authority for the corresponding as-machine resource policy to define a machine.

### Syntax

This subcommand has the following format:

```
insert_machine: machine_name
machine_name
```

Defines a unique name for the machine definition. When defining jobs, specify this name in the machine attribute.

**Limits:** Up to 80 alphanumeric characters; the value cannot contain embedded blanks or tabs

**Note:** If you do not specify the node\_name attribute in your machine definition, the node name is set to this *machine\_name* by default. In this situation, the *machine\_name* must be the DNS name of the agent machine. Otherwise, CA Workload Automation AE cannot connect to the agent.

### Required Attributes

To define a CA Workload Automation Agent for z/OS machine, you must specify the following attribute after the `insert_machine` subcommand:

- [`opsys: zos`](#) (see page 833)

If you are defining a CA Workload Automation Agent for Windows machine and the jobs will run under users that belong to a Windows network domain, you must specify the following attribute:

- [`opsys: windows`](#) (see page 833)

### Optional Attributes

You can specify the following optional attributes when defining a machine:

- [`description`](#) (see page 824)
- [`factor`](#) (see page 826)
- [`machine`](#) (see page 830)
- [`max\_load`](#) (see page 831)
- [`node\_name`](#) (see page 833)
- [`type`](#) (see page 837)

If you are defining a CA Workload Automation Agent for UNIX, Linux, Windows, i5/OS, or z/OS machine, the following attributes are also optional:

- [`agent\_name`](#) (see page 821)
- [`character\_code`](#) (see page 823)
- [`encryption\_type`](#) (see page 824)
- [`key\_to\_agent`](#) (see page 829)
- [`opsys`](#) (see page 833)
- [`port`](#) (see page 835)

#### **Example: Define an r11 Real Machine**

This example defines a legacy (r11) real Windows machine named plato:

```
insert_machine: plato
type: n
```

#### **Example: Define an r11.3 Real Machine**

This example defines an r11.3 real Windows machine named cheetah:

```
insert_machine: cheetah
type: a
opsys: windows
```

#### **Example: Define a Virtual Machine to Include Two Real Machines**

This example defines a virtual machine virtual\_b to include two real machines (france and italy) without specifying factors or loads:

```
insert_machine: virtual_b
type: v
machine: france
machine: italy
```

**Example: Define Two Real Machines and a Virtual Machine with Load Balancing Criteria**

This example defines two real Windows machines (elm and maple) and a virtual machine (trees) that includes both the real machines:

```
insert_machine: elm
type: n max_load: 100 factor: 1.0
insert_machine: maple
type: n max_load: 200 factor: 1.0
insert_machine: trees
type: v
machine: elm
max_load: 50 factor: .1
machine: maple
max_load: 100 factor: 1.0
```

**Example: Define a Real Machine Pool to Include Three Real Machines**

This example defines a real machine pool DCAPPOOL to include three real machines (MWIN, MLIN, and MSOL) that are discovered and monitored by CA Spectrum Automation Manager for real time load balancing:

```
insert_machine: DCAPPOOL
type: p
machine: MWIN
machine: MLIN
machine: MSOL
```

## Real Machines

*Real machines* are the definitions of actual machines where an agent has been installed.

You can use the `max_load` attribute in a real machine specification to define how many load units are allowed on that machine simultaneously. You can also use the `factor` attribute to specify the relative processing capacity of the machine. Both of these attributes are assigned from an arbitrary, user-defined range of values. `max_load` is used with the `job_load` attribute to limit the load placed on a machine at any one time. To avoid overloading a machine based on these attribute values, the scheduler queues jobs to run when load units become available again.

When you specify multiple real machines for a virtual machine, the scheduler determines which of the real machines have sufficient load units to run the job. If more than one real machine has sufficient units to run the job, the scheduler queries each machine's available CPU cycles, multiplies that number by the machine's factor, and chooses the machine with the largest value.

**Notes:**

- For more information about real machines, load balancing, and how CA Workload Automation AE selects a machine, see the *User Guide*.
- For more information about implementing load balancing using resources and real machine pools, see the *User Guide*.

## Virtual Machines

*Virtual machines* are the machine definitions that reference one or more existing real machine definitions. Virtual machines are typically used for load balancing, but they can be used for other things, such as creating an alias for a machine. For example, you can define a virtual machine named "produnix", which references the real name of a production UNIX machine. If job definitions reference the virtual machine name, the production UNIX machine can be changed in the virtual machine definition without having to change all jobs that referenced it.

If CA Workload Automation AE is running in external security mode using CA EEM, you must have CREATE authority for the corresponding as-machine resource policy to define the virtual machine.

To define a virtual machine, follow the `insert_machine` subcommand with one or more machine attributes to specify the real machines it contains, as follows:

```
insert_machine: virtual_machine_name
machine: real_machine_name
machine: real_machine_name
.
.
.
```

The following attributes can be added to the definition:

- [description](#) (see page 824)
- [factor](#) (see page 826)
- [max\\_load](#) (see page 831)
- [type](#) (see page 837)

**Note:** For more information about virtual machines, load balancing, and how CA Workload Automation AE selects a machine, see the *User Guide*.

## Real Machine Pools

Real machine pools (type "p" machine) are the machine definitions that reference one or more existing real machine definitions monitored by CA Spectrum Automation Manager. Real machine pools are typically used for load balancing by CA Spectrum Automation Manager.

To define a real machine pool, follow the insert\_machine subcommand with one or more machine attributes to specify the real machines it contains, as follows:

```
insert_machine: real_machine_pool_name
type: p
machine: real_machine_name
machine: real_machine_name
.
.
.
```

The [description](#) (see page 824) attribute can be added to the machine definition.

**Note:** For more information about implementing load balancing using real machine pools, see the *User Guide*.

## update\_machine Subcommand—Update a Machine Definition

The update\_machine subcommand updates a machine definition in the CA Workload Automation AE database.

**Note:** If CA Workload Automation AE is running in external security mode using CA EEM, you must have WRITE authority for the corresponding as-machine resource policy to update a machine.

### Syntax

This subcommand has the following format:

update\_machine: *machine\_name*

#### *machine\_name*

Specifies the name of the machine to update.

**Limits:** Up to 80 alphanumeric characters; the value cannot contain embedded blanks or tabs

### Optional Attributes

You can specify the following optional attributes when updating a machine:

- [description](#) (see page 824)
- [factor](#) (see page 826)
- [machine](#) (see page 830)
- [max\\_load](#) (see page 831)
- [node\\_name](#) (see page 833)

If you are updating a CA Workload Automation Agent for UNIX, Linux, Windows, i5/OS, or z/OS machine, the following attributes are also optional:

- [agent\\_name](#) (see page 821)
- [character\\_code](#) (see page 823)
- [encryption\\_type](#) (see page 824)
- [key\\_to\\_agent](#) (see page 829)
- [opsys](#) (see page 833)
- [port](#) (see page 835)

## Updating Virtual Machines

If CA Workload Automation AE is running in external security mode using CA EEM, you must have WRITE authority for the corresponding as-machine resource policy to update the virtual machine.

If you are updating a virtual machine that contains real machines, only the following attributes can be updated:

- [description](#) (see page 824)
- [factor](#) (see page 826)
- [max\\_load](#) (see page 831)

## agent\_name Attribute—Specify the Name of an Agent

The agent\_name attribute specifies the name of an agent.

**Notes:**

- This attribute does not apply to the legacy agent.
- If you do not specify the agent\_name attribute in your machine definition, the agent name is set to WA\_AGENT (the default).

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

## Syntax

This attribute has the following format:

agent\_name: *agent*

***agent***

Specifies the name of an agent. This name must match *one* of the following values:

- The agentname parameter in the agent's agentparm.txt file (on CA WA Agent for UNIX, Linux, Windows, or i5/OS)
- The ZOSAGENT(*name*) parameter in the AGENTDEF data set (on CA WA Agent for z/OS)

**Default:** WA\_AGENT

**Limits:** Up to 16 characters

**Note:** For more information about the agent parameters, see the *Implementation Guide* for your agent.

## character\_code Attribute—Specify the Character Code for the Machine

The character\_code attribute specifies the character code for an agent.

### Notes:

- This attribute does not apply to the legacy agent.
- If you do not specify the character\_code attribute in your job definition, the machine is defined with the ASCII character code (the default).

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

character\_code: ASCII | EBCDIC

#### ASCII

Specifies the ASCII character code. Specify this value if you are defining a CA WA Agent for UNIX, Linux, Windows, or i5/OS instance. This is the default.

#### EBCDIC

Specifies the EBCDIC character code. Specify this value if you are defining a CA WA Agent for z/OS instance.

## description Attribute—Specify a Description for an Agent

The description attribute specifies a description for the agent you are defining.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

`description: text`

***text***

Specifies a description for the agent.

**Limits:** Up to 256 characters

## encryption\_type Attribute—Specify the Type of Encryption

The encryption\_type attribute specifies the type of encryption between CA Workload Automation AE and the agent machine you are defining.

### Notes:

- This attribute does not apply to the legacy agent.
- If you do not specify the encryption\_type attribute in your machine definition, the machine uses DEFAULT encryption (the default).

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

## Syntax

This attribute has the following format:

encryption\_type: NONE | DEFAULT | AES

### NONE

Specifies that the machine uses no encryption.

### DEFAULT

Specifies that the machine uses the default encryption key and type. This is the default.

### AES

Specifies that the machine uses AES 128-bit encryption.

**Note:** You must specify a key using the key\_to\_agent attribute.

**CA WA Agent for z/OS Note:** To use encryption, both the agent on z/OS and CA Workload Automation AE must support AES encryption. You must specify the same CA Workload Automation AE encryption setting in the AGENTDEF data set of the agent. For more information about the encryption types that the agent supports, see the CA WA Agent for z/OS documentation.

## factor Attribute—Define a Factor for an Agent Machine

The factor attribute defines a factor to multiply by a real machine’s available CPU cycles to determine the “relative available CPU cycles.” When more than one real machine or a virtual machine is specified in the job’s machine attribute, the *real\_number* value determines on which machine a job should run. The factor value indicates a machine’s relative processing power. For example, a small machine might have a value of 0.2, while a powerful machine might have a value of 1.0. These values are arbitrary; any range can be used.

You can use the factor machine attribute **only** in real and virtual machine definitions. The factor machine attribute does not support machines of type ‘c’ or ‘u’.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

**factor: *real\_number***

#### ***real\_number***

Defines a real number from a user selected range of values.

#### **Limits:**

- 0-1024; up to 7 characters; you can specify a decimal.
- The number of decimal places cannot exceed hundredths (for example, 1023.99 is valid, but 1.999 is not valid).
- The examples in the following section use a range of 0.0 to 1.0; however, any reasonable convention is valid. For example, 100.00 to 999.99 is a valid range, but -1.0 and 1024.01 are not valid.

#### **Default:** 1.0

**Example: Set the Factor for a Very High-Performance Real Machine**

This example sets the factor for a very high-performance real machine on a scale of 0.0 to 1.0:

```
factor: 1.0
```

**Note:** Real machines and virtual machines can have different max\_load and factor values. The scheduler handles the load balancing according to the machine specified in the job definition. If the virtual machine is specified, the values associated with the virtual machine definition are used. If a real machine is specified, the values associated with the corresponding real machine definition are used.

**Example: Set the Factor for a Relatively Low-Performance Real Machine**

This example sets the factor for a relatively low-performance real machine on a scale of 0.0 to 1.0:

```
factor: 0.4
```

Assume the following virtual machines are defined:

```
insert_machine: marmot
machine: cheetah
factor: 1
machine: hippogriff
factor: .8
machine: goat
factor: .3
```

If the machine attribute for a job that is ready to start is set to the virtual machine marmot, the scheduler performs the necessary calculations to determine on which machine to run the job and reports these calculations in its output log. For example:

```
EVENT: STARTJOB JOB: test_mach
Checking Machine usages using RSTATD
:<cheetah=78*[1.00]=78> <hippogriff=80*[.80]=64>
<goat=2*[.30]=0.6>
[cheetah connected]
EVENT: CHANGE_STATUS STATUS: STARTING JOB:
test_mach
```

The factors weigh each machine to account for variations in processing power. In this example, even though the raw available CPU for cheetah was less than that of hippogriff, the product chose to run the job on cheetah based on its factor calculation:

```
(78 * 1.0 > 80 * 0.8)
```

That is, cheetah had more relative CPU cycles available.

When max\_load attributes are specified for the real machines in a virtual machine, the product determines which of the real machines have sufficient load units to run the job. If more than one real machine has sufficient units to run the job, the scheduler queries each machine's available CPU cycles, multiplies that number by the machine's factor, and chooses the machine with the largest value.

## force Attribute—Force a Machine to be Deleted

The force attribute deletes the machine whether it is currently being used or not.

### Supported JIL Subcommand

This attribute is optional for the [delete\\_machine subcommand](#) (see page 809).

### Syntax

This attribute has the following format:

force: y | n

y

Deletes the machine whether it is currently being used or not.

n

Does not delete the machine if it is currently being used. This is the default.

**Note:** After you delete the machine, its jobs, containers (such as virtual machines), and resources are no longer associated with any machine. You can re-insert the machine to associate the objects with it.

### Example: Force a Machine to be Deleted

This example deletes the real machine named realmach1, which is currently being used.

```
delete_machine: realmach1
force: y
```

## key\_to\_agent Attribute—Specify the Agent Encryption Key

The key\_to\_agent attribute specifies the key used to encrypt data from CA Workload Automation AE to the agent.

**Note:** This attribute only applies to the CA Workload Automation Agents.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

`key_to_agent: key`

#### ***key***

Specifies the key used to encrypt data from CA Workload Automation AE to the agent. This value must match the security.cryptkey parameter in the agent's agentparm.txt file, without the prefix 0x. If the values do not match, CA Workload Automation AE cannot communicate with the agent. You must specify one of the following:

- A 32-digit hexadecimal key
- A passphrase with up to 16 characters

## machine Attribute—Define or Update a Real Machine As a Component of the Virtual Machine or Real Machine Pool

The machine attribute defines a real machine as a component of the virtual machine or real machine pool that is being defined or updated by the `insert_machine` subcommand. If the specified virtual machine or real machine pool does not exist, it is created.

**Note:** This attribute differs from the machine attribute used in the job definition subcommands. In a job definition, the machine attribute assigns the job to one or more real or virtual machines.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [delete\\_machine](#) (see page 809)
- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

`machine: machine_name`

#### *machine\_name*

Specifies a real machine as a component of the virtual machine or real machine pool. The specified machine must have been defined to CA Workload Automation AE as a real machine.

**Limits:** Up to 80 alphanumeric characters; the value cannot contain embedded blanks or tabs

**Note:** The type of the specified machine cannot be v, w, or p. You can define virtual resources for any real machine defined on CA Workload Automation AE.

### Example: Define a Virtual Machine to Include Two Real Machines

This example defines a virtual machine `virtual_a` to include two real machines (`test` and `prod`). If the virtual machine already exists, the component machines are inserted.

```
insert_machine: virtual_a
machine: test
machine: prod
```

## max\_load Attribute—Define the Maximum Load an Agent Machine Can Handle

The max\_load attribute defines the maximum load (in load units) that a machine can reasonably handle. Load units are arbitrary values. The range of values is user-defined.

You can assign a max\_load value to each machine to indicate its relative processing capacity. For example, you might define that a powerful machine can handle up to 100 load units while a less powerful machine can handle only 10 load units. Similarly, you can use the job\_load attribute to assign values to your jobs that indicate how much relative processing power the jobs require. When a job is ready to run, the product compares the available max\_load value for each machine with the job\_load value for the job to determine which machine is the best fit for the job.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

`max_load: load_units`

#### *load\_units*

Defines how many load units are allowed on the machine simultaneously. This number can be any value in the user-defined range of possible values. The range is also arbitrary.

**Limits:** 0-999999999

#### Notes:

- Set the value to 0 for unlimited load units.
- If you do not set a max\_load value, CA Workload Automation AE does not limit the load on the machine.

**Notes:**

- If job\_load is not set for a job, the job runs without checking for load units. If a priority is not set for a job, the priority defaults to 0 and the job\_load value is ignored.
- When you specify multiple machines in the job's machine attribute, each machine is checked for available load units when the job is ready to run. If none of the machines currently has the necessary load units available, the job is queued on all of the specified machines. The job will be submitted to the first machine where the necessary load units are available.
- The job\_load attribute is not related to the priority or ordering of jobs in a queue. The job\_load attribute only controls the machine where the job runs when it exceeds the max\_load of a machine, thus eliminating that machine.
- You can use the max\_load attribute in real and virtual machine definitions. Real machines and virtual machines can have different max\_load and factor values. The scheduler handles the load balancing according to the machine specified in the job definition. If the virtual machine is specified, the values associated with the virtual machine definition are used. If a real machine is specified, the values associated with the corresponding real machine definition are used.

**Example: Set the Maximum Load for a Very High-Performance Real Machine**

This example sets the max\_load for a very high-performance real machine on a scale of 1 to 100.

```
max_load: 100
```

**Example: Set the Maximum Load for a Relatively Low-Performance Real Machine**

This example sets the max\_load for a relatively low-performance real machine on a scale of 1 to 100.

```
max_load: 20
```

## node\_name Attribute—Specify the IP Address of a Machine

The node\_name attribute specifies the IP address or DNS name of the machine you are defining.

**Note:** If you do not specify the node\_name attribute in your machine definition, the value specified in the insert\_machine: *machine\_name* command is used (the default).

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

node\_name: *address*

#### *address*

Specifies the IP address or DNS name of the machine.

**Default:** The value specified in the insert\_machine: *machine\_name* command. In this situation, the *machine\_name* must be the DNS name of the agent machine. Otherwise, CA Workload Automation AE cannot connect to the agent.

**Limits:** Up to 80 characters

**Example:** 172.24.36.107 (IPv4) or 0:0:0:0:FFFF:192.168.00.00 (IPv6)

## opsys Attribute—Specify the Operating System Type of a Machine

The opsys attribute specifies the operating system type of the machine you are defining.

**Note:** This attribute only applies to CA Workload Automation Agent machines. To use this attribute, you must specify the type: *a* attribute.

### Supported JIL Subcommands

This attribute is optional for the following subcommands if the type attribute is set to *a*:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

## Syntax

This attribute has the following format:

`opsys: type`

### ***type***

Specifies the operating system type of the machine. Options are the following:

- aix
- hpx
- linux
- openvms
- i5os
- solaris
- tandem
- windows
- zos

### **Notes:**

- If the machine is running on Windows and the jobs run under users that belong to a Windows network domain, you must add the `opsys: windows` attribute to the machine definition.
- If you specify `opsys: zos` in your machine attribute, you must also specify `encryption_type: NONE` or `encryption_type: AES`.

## port Attribute—Specify the Port Used to Communicate with the Machine

The port attribute specifies the port that the agent uses to listen for traffic.

### Notes:

- This attribute does not apply to the legacy agent.
- If you do not specify the port attribute in your machine definition, the machine is defined with port 7520 (the default).

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_machine](#) (see page 813)
- [update\\_machine](#) (see page 820)

### Syntax

This attribute has the following format:

`port: port_number`

#### *port\_number*

Specifies the port that the agent uses to listen for traffic. This name must match *one* of the following values:

- The communication.input.port parameter in the agent's agentparm.txt file (on CA WA Agent for UNIX, Linux, Windows, or i5/OS)
- The COMMCHAN PORT(port) parameter in the AGENTDEF data set (on CA WA Agent for z/OS)

**Default:** 7520

**Limits:** 1-65535

**Note:** For more information about the agent parameters, see the Implementation Guide for your agent.

## remove\_references Attribute—Remove All Machine References from Virtual Machines or Real Machine Pools

The remove\_references attribute removes all references to a real machine and deletes the machine. This attribute only applies to real machines.

**Note:** The remove\_references attribute does not delete the real machine if it is being referenced by virtual resources or active jobs.

### Supported JIL Subcommand

This attribute is optional for the [delete\\_machine subcommand](#) (see page 809).

### Syntax

This attribute has the following format:

remove\_references: y | n

y

Removes all references from virtual machines or real machine pools and deletes the real machine definition.

n

Does not remove all references from virtual machines or real machine pools and does not delete the real machine definition. This is the default.

### Example: Delete a Real Machine and Remove All References to It

This example deletes the real machine named realmach1 and removes all references to it from virtual machines or real machine pools.

```
delete_machine: realmach1
remove_references: y
```

## type Attribute—Specify the Machine Type

The type attribute specifies the type of machine you are defining. The type attribute is optional in a machine definition.

**Note:** If you do not specify the type attribute in your machine definition, the machine is defined as type a (the default).

### Supported JIL Subcommand

This attribute is optional for the [insert machine subcommand](#) (see page 813).

### Syntax

This attribute has the following format:

type: a | c | l | L | n | p | r | u | v

#### a

Specifies a CA Workload Automation Agent for UNIX, Linux, Windows, i5/OS, or z/OS machine. This is the default.

#### c

Specifies a CA AutoSys Workload Automation Connect Option machine.

#### l

Specifies a 4.5 real UNIX machine. You must specify a lowercase l.

#### L

Specifies a 4.5 real Windows machine. You must specify a capital L.

#### n

Specifies an r11 real Windows machine or a virtual machine that consists only of r11 real Windows machines (type n).

#### p

Specifies a real machine pool managed by CA Spectrum Automation Manager.

**Note:** In the documentation, the type "p" machine is referred to as the real machine pool.

#### r

Specifies an r11 real UNIX machine.

#### u

Specifies a CA NSM or a CA Universal Job Management Agent (CA UJMA) machine.

v

Specifies a virtual machine. The virtual machine can consist of CA Workload Automation Agent machines (type a), r11 real UNIX machines (type r), and r11 real Windows machines (type n).

**Notes:**

- With r11.3, the scheduler component has the capability to run jobs on the r11.3 agent machines, r11 agent machines, and r4.x remote agent machines. Due to the differences in the communication protocol used by the different versions of the agents, the scheduler component must know which communication protocol to invoke before contacting the agent machine. This is done by examining the agent's machine definition type attribute.
- In addition to setting the type attribute of the machine definition for each r4.x remote agent machine, the Legacy Remote Agent Port value must also be set in the CA Workload Automation AE environment. This value specifies the port number on which the scheduler communicates with r4.x remote agents.
- On Windows, the Legacy Remote Agent Port value can be set in the Scheduler - CA Workload Automation AE Administrator window of CA Workload Automation AE Administrator. On UNIX, you can set this value by locating and updating the AutoRemPort string within the \$AUTOUSER/config.<instance> file.
- The machine types "t" and "z" available in the r4.x releases of the product are now replaced with "u" and "c" respectively. Using JIL, if you input the machine type as "t" it is converted to "u" and if you input the machine type as "z" it is converted to "c", and a warning message appears.
- Suppose that you define an r11 virtual machine with type "v" and it only contains r11 real Windows machines (type "n"), as shown in the following example:

```
insert_machine: realmach1
type: n
insert_machine: realmach2
type: n
insert_machine: virtmach
type: v
machine: realmach1
machine: realmach2
```

Because the container machine (virtmach) only contains Windows real machines, its type is automatically changed from type "v" to type "n" in the database. This change indicates that virtmach is a Windows virtual machine. If you run the autorep command to display the virtual machine, type: n is reported. This change only applies to the legacy agent.

# Chapter 7: JIL Resource Definitions

---

This chapter describes the JIL subcommands and attributes that you can use to define, update, and delete resources. To define, update, or delete a resource, you specify the appropriate JIL subcommand and attribute statements to the `jil` command.

## **delete\_resource Subcommand—Delete a Virtual Resource**

The `delete_resource` subcommand deletes a virtual resource definition from the database.

**Note:** You cannot delete a virtual resource if it is referenced as a dependency in a job. To delete the resource, you must delete all the related job dependencies first. If a resource is deleted while a job that references it as a dependency is active or running, the job continues to run and is not affected.

### Syntax

This subcommand has the following format:

```
delete_resource: resource_name
resource_name
```

Specifies the name of the virtual resource that you want to delete.

### Optional Attribute

To delete a virtual resource, you can specify the following optional attribute following the `delete_resource` subcommand:

- [machine](#) (see page 473)

### Example: Delete a Global Virtual Resource

This example deletes the global virtual resource named `glob_resource`.

```
delete_resource: glob_resource
```

### Example: Delete a Machine-Level Virtual Resource

This example deletes the virtual resource named `mach_resource` that is associated with the `winagent` machine.

```
delete_resource: mach_resource
machine: winagent
```

## insert\_resource Subcommand—Define a Virtual Resource

The insert\_resource subcommand adds a virtual resource definition to the database. A virtual resource is representational only and is not directly tied to a physical system. Using virtual resources helps you control job execution and to improve your environment's performance.

For example, you can define a virtual resource to represent the maximum number of floating product licenses available in your enterprise. Each time a qualified job runs, a unit of that resource is used. When all the units are used, no more jobs can run.

**Note:** You can define resources on distributed machines only. Before you can define a resource on a machine, the machine must already be defined on the database.

### Syntax

This subcommand has the following format:

`insert_resource: resource_name`  
***resource\_name***

Defines the name of the virtual resource.

**Limits:** Up to 128 characters; can include a-z, A-Z, 0-9, period (.), underscore (\_), hyphen (-), and pound (#); cannot include embedded spaces or tabs

**Note:** The virtual resource name must be unique across all resource types.

### Notes:

- You cannot define the same virtual resource at the machine level and global level.
- You can define the same virtual resource on multiple machines.

### Required Attributes

To define a virtual resource, you must specify the following attributes following the insert\_resource subcommand:

- [amount](#) (see page 844)
- [res\\_type](#) (see page 847)

### Optional Attributes

You can specify the following optional attributes when defining a virtual resource:

- [description](#) (see page 845)
- [machine](#) (see page 473)

### Example: Define a Virtual Resource at the Machine Level

This example defines a renewable resource named renew\_res1 for the unixagent1 machine. The resource is assigned 10 units.

```
insert_resource: renew_res1
res_type: R
machine: unixagent1
amount: 10
```

Suppose that you issue the following subcommand after adding the previous resource. The machine attribute is not specified, so the subcommand tries to add a renewable resource also named renew\_res1 at the global level. Machine-level and global-level resources cannot have the same name, so the subcommand fails.

```
insert_resource: renew_res1
res_type: R
amount: 25
```

### Example: Define the Same Virtual Resource on Two Machines

This example defines renewable virtual resources on machine1 and machine2 with the same name. The resources are successfully created because you can define the same resource on multiple machines if the res\_type value is also the same.

```
insert_resource: renew_count
machine: machine1
res_type: R
amount: 10
```

```
insert_resource: renew_count
machine: machine2
res_type: R
amount: 20
```

Suppose that the following job is defined to run on a virtual machine named virtmach, which contains machine1 and machine2. The renew\_count resource is defined as a dependency, as follows:

```
insert_job: res_dep_job
job_type: CMD
machine: virtmach
command: /u1/procrun.sh
resources: (renew_count, QUANTITY=5, free=Y)
```

CA Workload Automation AE searches both machine1 and machine2, chooses the "best" machine that satisfies the job's resource dependency, and runs the job on that machine. If both machines satisfy the resource requirements, the job runs on the first machine.

**Note:** A job can have both virtual and real resource dependencies. In this scenario, the resource dependencies are evaluated as follows:

- CA Workload Automation AE gathers machines that satisfy the virtual resource dependencies.
- CA Workload Automation AE calls CA Spectrum Automation Manager and communicates the list of satisfied machines along with the real resource dependencies.
- CA Spectrum Automation Manager chooses the "best" machine.
- The job runs on that machine.
- If CA Workload Automation AE is not integrated with CA Spectrum Automation Manager, the real resource dependencies are ignored and the job is submitted on the first machine that satisfies the virtual resource requirements.

## update\_resource Subcommand—Update a Virtual Resource

The update\_resource subcommand updates the amount and description properties of a virtual resource definition in the database.

**Note:** You cannot update resource types or machine names using the update\_resource subcommand. To update resource types or machine names, you must delete the resource and add it to database again with the new properties.

### Syntax

This subcommand has the following format:

`update_resource: resource_name`

#### *resource\_name*

Defines the name of the virtual resource that you want to update. This resource must be defined in the database.

### Optional Attributes

To update a virtual resource, you can specify the following optional attributes following the update\_resource subcommand:

- [amount](#) (see page 844)
- [description](#) (see page 845)

To update a virtual resource that is associated with a machine, you must specify the following attribute:

- [machine](#) (see page 473)

#### Example: Update a Virtual Resource that is Associated with a Machine

This example updates a virtual resource that is associated with the unixagent machine. The number of units is changed to 35.

```
update_resource: mach_res
machine: unixagent
amount: 35
```

Suppose that the amount attribute is defined as follows:

```
amount: +20
```

In this situation, the command adds 20 units to the available resource count. For example, if the resource already has 35 units, the command adds 20 units and the total would be 55.

Similarly, suppose that the amount attribute is defined as follows:

```
amount: -20
```

If the resource has 35 units, the command removes 20 units and the total would be 15.

#### Example: Update a Global Virtual Resource

This example defines a global virtual resource and assigns 10 units to it.

```
insert_resource: glob_res
res_type: D
amount: 10
```

This following command updates the same resource and assigns 25 units to it:

```
update_resource: glob_res
amount: 25
```

## amount Attribute—Specify the Amount of Virtual Resource

The amount attribute specifies the number of units to assign to a virtual resource.

### Supported JIL Subcommands

This attribute is required for the [insert\\_resource subcommand](#) (see page 840).

This attribute is optional for the [update\\_resource subcommand](#) (see page 842).

### Syntax

This attribute has the following format:

amount: *value*

#### *value*

Defines the number of units to assign to the virtual resource.

**Limits:** Integer; up to 2000000000

**Note:** If you are updating an existing resource, you can specify an absolute number or a relative number (for example, +3 or -5). Specifying a relative number lets you increase or decrease the amount without assigning a specific number. The amount cannot be decremented to a negative value.

### Example: Update a Global Virtual Resource Using an Absolute Amount

This example assigns 30 units to the count\_res resource. The machine attribute is not specified, so the count\_res resource must be a global resource.

```
update_resource: count_res
amount: 30
```

### Example: Update a Machine-Level Virtual Resource Using a Relative Amount

This example updates the win\_res resource that is associated with the winagent machine. The amount is increased by 5 units.

```
update_resource: win_res
machine: winagent
amount: +5
```

## description Attribute—Specify a Description for a Virtual Resource

The amount attribute specifies a description for a virtual resource.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_resource](#) (see page 840)
- [update\\_resource](#) (see page 842)

### Syntax

This attribute has the following format:

```
description: "text"
"text"
```

Defines a free-form text description for the virtual resource.

**Limits:** Up to 256 alphanumeric characters (including spaces)

### Example: Specify a Description for a Virtual Resource

This example defines a new resource that includes a description.

```
insert_resource: renew_res
machine: winagent
res_type: R
amount: 10
description: "This renewable resource is only available for the winagent machine."
```

You can issue the following update\_resource subcommand to change the amount and description:

```
update_resource: renew_res
machine: winagent
amount: 25
description: "This renewable resource is only available for the winagent machine. The number of units is 25."
```

## machine Attribute—Specify the Machine for a Virtual Resource

The machine attribute specifies the machine name that a virtual resource is associated with. A resource defined for a specific machine is only available to jobs submitted to that machine.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [delete\\_resource](#) (see page 839)
- [insert\\_resource](#) (see page 840)
- [update\\_resource](#) (see page 842)

#### Notes:

- If you omit the machine attribute from a new virtual resource definition, the resource is added as a global resource that is available to all machines.
- You cannot update the machine name by specifying the machine attribute with the update\_resource subcommand. To update a machine name, you must delete the resource and add it to the database again with the same name.
- You can specify the machine attribute with the update\_resource or delete\_resource subcommands to identify the virtual resource you want to update or delete.

### Syntax

This attribute has the following format:

machine: *machine\_name*

#### *machine\_name*

Specifies the name of the machine that the virtual resource is defined for.

**Limits:** Up to 80 characters; can include a-z, A-Z, 0-9, period (.), underscore (\_), hyphen (-), and pound (#); cannot include embedded spaces or tabs

#### Notes:

- The machine must already be defined to CA Workload Automation AE.
- The type of the specified machine cannot be v, w, or p. You can define virtual resources for any machine defined on CA Workload Automation AE.
- The specified machine cannot be a virtual machine or real machine pool.

### Example: Define a Global Virtual Resource

This example defines a virtual depletable resource named glob\_res. The machine attribute is not specified, so the resource is available to all machines.

```
insert_resource: glob_res
res_type: D
amount: 50
description: "This resource is permanently consumed."
```

### Example: Update a Resource Amount

Suppose that threshold\_res is an existing resource on the prodserv machine and you issue the following update\_resource subcommand:

```
update_resource: threshold_res
amount: 25
```

The machine attribute is not specified, so the subcommand tries to update a global resource named threshold\_res, which does not exist. The subcommand fails.

To update the threshold\_res resource, you must include the machine attribute in the update\_resource subcommand as follows:

```
update_resource: threshold_res
machine: prodserv
amount: 25
```

## res\_type Attribute—Specify the Type of Virtual Resource

The res\_type attribute specifies the type of virtual resource.

### Supported JIL Subcommand

This attribute is required for the [insert\\_resource subcommand](#) (see page 840).

### Syntax

This attribute has the following format:

res\_type: D | R | T

#### D

Defines a depletable resource. A depletable resource is a consumed resource. When a job that uses this resource is submitted, the used resource units are permanently removed from the available resource pool. If you want to use the depletable resource again in a job, you must use the update\_resource subcommand to replenish the resource units back to the resource pool.

**R**

Defines a renewable resource. A renewable resource is a borrowed resource. When a job that uses this resource is submitted, the used resource units are temporarily removed from the available resource pool. When the job completes, the resource units are returned to the pool, or the units are held until they are manually released back to the pool. You specify whether the units are returned or not when you define the resource dependency in a job.

**T**

Defines a threshold resource. A threshold resource is a sizing resource. Only the jobs that require the specified resource amount or fewer are submitted to run. For example, if the threshold resource is set to 2, CA Workload Automation AE submits the jobs that require 2 or fewer units. The used resource units are not removed from the resource pool.

**Note:** You cannot update a resource's type using the `update_resource` subcommand. To update a resource's type, you must delete the resource and add it to database again with the new type.

**Example: Define a Global Renewable Resource**

This example defines a depletable resource that is available to all machines. The resource is assigned 10 units.

```
insert_resource: renewable_res
res_type: R
amount: 10
```

Suppose that the `renewable_res` resource is defined as a dependency to the following job:

```
insert_job: res_dep_job
job_type: CMD
machine: unixagent
command: /u1/procrun.sh
resources: (renewable_res, QUANTITY=5, FREE=Y)
```

When this job is submitted, five resource units are temporarily removed from the resource pool. If the job completes successfully, the resource units are returned to the pool. If the job does not complete successfully, the units are held until they are manually released back to the resource pool.

### Example: Define a Machine-Level Depletable Resource

This example defines a depletable resource for the winagent machine. The resource is assigned 25 units. Units that are used by jobs are permanently removed from the available resource pool.

```
insert_resource: depletable_res
res_type: D
amount: 25
```

### Example: Define a Machine-Level Threshold Resource

This example defines a threshold resource for the unixagent machine. The resource is assigned 10 units, so only jobs that require 10 or fewer units of this resource are submitted to the unixagent machine.

```
insert_resource: threshold_res
res_type: T
machine: unixagent
amount: 10
```

## resources Attribute—Define or Update Real and Virtual Resource Dependencies in a Job

The resources attribute defines or updates real and virtual resource dependencies in a job.

### Notes:

- To define a virtual resource dependency, the virtual resource must already be defined on the database. Otherwise, you get an error.
- You cannot define a resource dependency on a box.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_job](#) (see page 289)
- [update\\_job](#) (see page 292)

## Syntax

This attribute has the following formats:

- To specify one or more virtual resource dependencies:  
resources: (*virtual\_resource\_name*, quantity=*amount*|ALL[, free=Y|N|A]) [AND ...]
- To specify one or more real resource dependencies:  
resources: (*real\_resource\_type*, VALUEOP=*operator*, VALUE=*res\_value*) [AND ...]
- To specify real and virtual resource dependencies:  
resources: (*virtual\_resource\_name*, quantity=*amount*|ALL[, free=Y|N|A])  
AND (*real\_resource\_type*, VALUEOP=*operator*, VALUE=*res\_value*)  
[AND ...]

### *virtual\_resource\_name*

Specifies the name of a virtual resource.

**Note:** The virtual resource must already be defined in the database.

### **quantity=***amount* | ALL

Specifies the amount of virtual resource required. Options are the following:

- *amount*—Specifies the number of units that the job requires. The number must be a positive integer.
- ALL—Specifies that the job requires all the units of the resource.

### **free=**Y | N | A

(Optional for renewable virtual resources only) Specifies whether the units of the virtual resource are freed from the job. Options are the following:

- Y—Frees the units only if the job completes successfully. This is the default.
- N—The units are not freed. To free the resources, you must issue the following command:  
`sendevent -E RELEASE_RESOURCE -J job_name`
- A—Frees the units unconditionally.

### *real\_resource\_type*

Specifies the type of the real resource. Options are the following:

#### CPU\_IDLE\_PCT

Defines the percentage of time over the sample period that the system's CPUs were idle.

**Corresponding metric in the CA SystemEDGE agent:** cpuTotalIdlePercent

**CPU\_LOAD\_AVG\_15MIN**

Defines the load average in the last 15 minutes.

**Corresponding metric in the CA SystemEDGE agent:** loadAverage15Min

**CPU\_LOAD\_AVG\_5MIN**

Defines the load average in the last 5 minutes.

**Corresponding metric in the CA SystemEDGE agent:** loadAverage5Min

**MEM\_INUSE\_PCT**

Defines the percentage of the system's active memory that is in use.

**Corresponding metric in the CA SystemEDGE agent:** memCapacity

**SWAP\_INUSE\_PCT**

Defines the percentage of the system's total swap that is in use.

**Corresponding metric in the CA SystemEDGE agent:** swapCapacity

**SWAP\_SPACE\_TOTAL**

Defines the total swap space (in KB).

**Corresponding metric in the CA SystemEDGE agent:** totalSwapSpace

**SYSTEM\_CPU\_COUNT**

Defines the total number of CPUs that the job requires.

**Corresponding metric in the CA SystemEDGE agent:** Number of CPUs

**SYSTEM\_CPU\_SPEED**

Defines the system clock speed (in MHz) the job requires.

**Corresponding metric in the CA ACM agent:** CPU Speed

**SYSTEM\_OS\_TYPE and VERSION**

Specifies the operating system name and version that the job requires.

**Corresponding metrics in the CA SystemEDGE agent:** OS Type/OS Version

**SYSTEM\_PHYSICAL\_MEMORY**

Defines the total amount of available physical memory (in MB) that the job requires.

**Corresponding metric in the CA SystemEDGE agent:** Physical Memory

### **SOFTWARE\_NAME and VERSION**

Specifies the software and version that the job requires.

**Corresponding metrics in the CA ACM agent:** SOFTWARE\_NAME/VERSION

**Note:** Real resources are predefined to CA Workload Automation AE and are managed by CA Spectrum Automation Manager. You cannot define or update real resources using CA Workload Automation AE.

### **VALUEOP=operator**

Specifies the operator for the real resource dependency. Options are the following:

- EQ
- NEQ
- LT
- LTE
- GT
- GTE

### **VALUE=res\_value**

Specifies the value of the real resource specified in the *real\_resource\_type* keyword.

**Note:** You can define multiple real and virtual resource dependencies in a job definition. Separate each dependency with **AND**.

### **Example: Define Real and Virtual Resource Dependencies**

This example defines a Command job that has real and virtual resource dependencies. Before the job can start running, it needs a machine that satisfies all the following dependencies:

- 1 unit of the depletable resource named D1
- 3 units of the threshold resource named T1
- 4 units of the renewable resource named R1
- SYSTEM\_OS\_TYPE is AIX 5.3
- SYSTEM\_PHYSICAL\_MEMORY is greater than 2 GB

```
insert_job: res_dep_job
job_type: CMD
machine: unixagent
command: /u1/procrun.sh
resources: (D1, QUANTITY=1) AND (T1, QUANTITY=3) AND
(R1, QUANTITY=4, FREE=Y) AND
(SYSTEM_OS_TYPE, VALUEOP=EQ, VALUE=AIX, VERSION=5.3) AND
(SYSTEM_PHYSICAL_MEMORY, VALUEOP=GT, VALUE=2097152)
```

## Considerations When Specifying the real\_resource\_type Option

The following are important considerations when you specify the real\_resource\_type option in the resources attribute:

- You must integrate CA Workload Automation AE to work with CA Spectrum Automation Manager to utilize the real resource constraints. Else, all the real resource constraints are ignored during job processing.
- The SysEdge agents provide performance data that is used by CA Spectrum Automation Manager during load balancing. You must install the SysEdge agents on all agent machines that utilize real resources for load balancing. If SysEdge is not running, the agent machine is not qualified for job submission although it satisfies the real resource constraints.
- The SYSTEM\_PHYSICAL\_MEMORY real resource is displayed in KB on the CA Spectrum Automation Manager user interface. You must divide this value by 1024 and approximate it to the next value in MB. For example, if the value displayed on CA Spectrum Automation Manager is 8175256 KB, you must approximate it to 7984 MB. You must use this approximated value when you define the real resource dependency.
- You must query the SYSTEM\_OS\_TYPE real resource using generic names like Windows, AIX, Linux, HPUX, and Solaris.
- You must query the SYSTEM\_OS\_VERSION real resource using generic versions 8, 9, 10 for SuSe Linux and 3, 4, 5 for RHEL.
- The SOFTWARE\_NAME and SOFTWARE\_VERSION resource values must match to what is discovered in CA ACM.
- Masks, wildcards, or regular expressions are not supported for the SYSTEM\_OS\_VERSION and SOFTWARE\_VERSION resource types.
- The ‘.’, ‘\_’, and ‘-’ delimiters are allowed.
- When you specify the SYSTEM\_OS\_VERSION or SOFTWARE\_VERSION real resource, the version is compared as follows:
  - String comparison is done in the case where either the actual or compared value contains any characters that are not delimiters or numbers.
  - If the values are all delimited numbers, each token is numerically compared. Leading zeros are dropped and the numbers are compared. For example, 5.2 is treated as less than 5.09. Both the values have the same first token, that is 5, and so they are equal. The second tokens are compared discarding the leading zeroes. So, the first value 5.2 is less than 5.09 because 2 less than 9.
- When you specify the SOFTWARE\_NAME real resource, it lets you use regular expressions pattern matching to search the names, as follows:
  - JRE version greater than 1.4.2\_04

**Example:** (SOFTWARE\_NAME, VALUE=JRE.\* , VALUEOP=GTE, VERSION=1.4.2\_04) indicates that the job requires a machine that has JRE version greater than or equal to 1.4.2\_04 installed.

- Any JRE version

**Example:** (SOFTWARE\_NAME, VALUE=J.\*[1-9].[0-9].\*) indicates that the job requires a machine that has any version of JRE installed.

- JRE 1.4 (UNIX) as obtained from CA ACM

**Example:** (SOFTWARE\_NAME, VALUE=JRE 1.4 \(\text{UNIX}\), VALUEOP=EQ, VERSION=1.4.2\_09) indicates that the job requires a machine that has JRE 1.4.2\_09 installed on UNIX.

- Sybase version 15.0.1

**Example:** (SOFTWARE\_NAME, VALUE=^Sybase.\* , VALUEOP=EQ, VERSION=15.0.1) indicates that the job requires a machine that has Sybase 15.0.1 installed.

- Any CA Workload Automation AE version

**Example:** (SOFTWARE\_NAME, VALUE=.\*CA Workload Automation AE.\* ) indicates that the job requires a machine that has any version of CA Workload Automation AE installed.

- CA Workload Automation AE agent version 11.0

**Example:** (SOFTWARE\_NAME, VALUE=.\*CA Workload Automation AE.\*Agent.\* , VALUEOP=EQ, VERSION=11.0) indicates that the job requires a machine that has CA Workload Automation AE agent 11.0 installed.

- Open SSL version 0.9.8

**Example:** (SOFTWARE\_NAME, VALUE=.\*SSL\$, VALUEOP=LTE, VERSION=0.9.8) indicates that the job requires a machine that has SSL 0.9.8 installed.

# **Chapter 8: JIL Cross-Instance Definitions**

---

This chapter describes the JIL subcommands and attributes that you can use to define, update, and delete external instances that CA Workload Automation AE can communicate with. An external instance can be another CA Workload Automation AE instance or it can be a CA Workload Automation product running on another platform, including mainframe. To define, update, or delete an external instance, you specify the appropriate JIL subcommand and attribute statements to the `jil` command.

## **delete\_xinst Subcommand—Delete an External Instance**

The `delete_xinst` subcommand deletes the specified external instance from the CA Workload Automation AE database.

### **Syntax**

This subcommand has the following format:

`delete_xinst: external_instance`  
***external\_instance***

Identifies the external instance to delete from the CA Workload Automation AE database.

**Limits:** Must be three alphanumeric characters; uppercase

### **Example: Delete an External Instance**

This example deletes the external instance ACE.

`delete_xinst: ACE`

## insert\_xinst Subcommand—Define an External Instance

The insert\_xinst subcommand defines a new external instance in the CA Workload Automation AE database. The external instance can be one of the following CA scheduling managers:

| Scheduling Manager          | Required Integration Software           | Environment |
|-----------------------------|-----------------------------------------|-------------|
| CA Job Management Option    | CA UJMA                                 | Distributed |
| CA Jobtrac Job Management   | CA AutoSys WA Connect Option or CA UJMA | Mainframe   |
| CA Scheduler Job Management | CA AutoSys WA Connect Option or CA UJMA | Mainframe   |
| CA Workload Automation AE   | None                                    | Distributed |
| CA Workload Automation EE   | None                                    | Mainframe   |
| CA Workload Automation SE   | CA AutoSys WA Connect Option or CA UJMA | Mainframe   |

### Syntax

This subcommand has the following format:

`insert_xinst: external_instance`  
***external\_instance***

Defines the unique name of the external instance to add.

**Limits:** Must be three alphanumeric characters

**Note:** Values specified in lowercase characters are converted to uppercase.

### Required Attributes

To define an external instance, you must specify the following attributes following the insert\_xinst subcommand:

- [xtype](#) (see page 865)
- [xmachine](#) (see page 862)

If you are defining an external CA Workload Automation AE instance, the following attribute is also required:

- [xport](#) (see page 864)

If you are defining an external CA Workload Automation EE instance, the following attributes are also required:

- [xmanager](#) (see page 863)
- [xport](#) (see page 864)

#### Optional Attributes

You can specify the following optional attributes when defining an external CA Workload Automation AE or CA Workload Automation EE instance:

- [xcrypt\\_type](#) (see page 860)
- [xkey\\_to\\_manager](#) (see page 861)

**Note:** If you are defining an external CA Workload Automation AE instance and it uses a shadow scheduler, you must add the xmachine and xport attributes for that shadow scheduler. You can also optionally add the xcrypt\_type and xkey\_to\_manager attributes.

#### Example: Define an External CA Workload Automation AE Instance

This example defines an external CA Workload Automation AE application server instance named ACE to run on machineA and port 9000 using the default encryption.

```
insert_xinst: ACE
 xtype: a
 xmachine: machineA
 xport: 9000
```

#### Example: Define Multiple External CA Workload Automation AE Instances

This example defines multiple external CA Workload Automation AE application server external instances named ACE to run on machineA and machineB at port 9000.

```
insert_xinst: ACE
 xtype: a
 xmachine: machineA,machineB
 xport: 9000
```

#### Example: Define an External CA AutoSys WA Connect Option Instance

This example defines an external CA AutoSys WA Connect Option instance named CA7 to run on the machine called MAINFR1.

```
insert_xinst: CA7
 xtype: c
 xmachine: MAINFR1
```

#### **Example: Define a CA Job Management Option External Instance**

This example defines an external CA Job Management Option instance named JMO to run on the machine called unimachA.

```
insert_xinst: JMO
xtype: u
xmachine: unimachA
```

## **update\_xinst Subcommand—Update an External Instance Definition**

The update\_xinst subcommand updates an existing external instance definition in the CA Workload Automation AE database. You can update the external instance definition when the connection information changes. The connection information must be accurate so that the scheduling managers can communicate with each other.

### **Syntax**

This subcommand has the following format:

`update_xinst: external_instance`

***external\_instance***

Identifies the external instance definition to update.

**Limits:** Must be three alphanumeric characters; uppercase

### **Optional Attributes**

To update an external instance, you can specify the following attribute following the update\_xinst\_subcommand:

- [xmachine](#) (see page 862)

To update an external CA Workload Automation AE instance, you can specify the following attributes:

- [xport](#) (see page 864)
- [xcrypt\\_type](#) (see page 860)
- [xkey\\_to\\_manager](#) (see page 861)

To update an external CA Workload Automation EE instance, you can specify the following attributes:

- [xmanager](#) (see page 863)
- [xport](#) (see page 864)
- [xcrypt\\_type](#) (see page 860)
- [xkey\\_to\\_manager](#) (see page 861)

## xcrypt\_type Attribute—Specify the Encryption Type

The xcrypt\_key attribute specifies the encryption type to use.

### Notes:

- This attribute only applies to CA Workload Automation AE (xtype: a) and CA Workload Automation EE (xtype: e) instances.
- If you do not specify this attribute, the instance is defined with the default encryption type for CA Workload Automation AE or no encryption for CA Workload Automation EE.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_xinst](#) (see page 856)
- [update\\_xinst](#) (see page 858)

### Syntax

This attribute has the following formats:

- To specify the encryption type for CA Workload Automation AE:  
xcrypt\_type: NONE | DEFAULT | AES
- To specify the encryption type for CA Workload Automation EE:  
xcrypt\_type: NONE | AES

### DEFAULT

Specifies that the default encryption key and type are used. This is the default for CA Workload Automation AE instances (xtype: a).

### NONE

Specifies that no encryption is used. This is the default for CA Workload Automation EE instances (xtype: e).

### AES

Indicates that the data is encrypted using AES 128-bit encryption. If you are using AES 128-bit encryption to encrypt the data, you must generate the instance-wide communication alias encryption file (cryptkey\_alias.txt) using the as-config command.

**Note:** When xcrypt\_type is set to AES, you must also specify the xkey\_to\_manager attribute.

**Note:** On CA Workload Automation EE, you must specify the same CA Workload Automation AE encryption setting in the AGENTDEF data set. Therefore, to use encryption, CA Workload Automation AE and the external instance must support the same encryption type.

## xkey\_to\_manager Attribute—Specify an Encryption Key for Manager Communication

The key\_to\_manager attribute specifies the encryption key used to encrypt data sent to the scheduling manager.

### Notes:

- This attribute only applies to CA Workload Automation AE and CA Workload Automation EE instances. Therefore, you must specify the xtype: a or xtype: e attribute.
- To use this attribute, you must specify AES for the xcrypt\_type attribute.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_xinst](#) (see page 856)
- [update\\_xinst](#) (see page 858)

### Syntax

This attribute has the following format:

`xkey_to_manager: key`

**key**

Specifies the key used to encrypt data between CA Workload Automation AE and the external instance. You must specify one of the following:

- A 32-digit hexadecimal key
- A passphrase with up to 16 characters (valid only for xtype: a)

**Note:** For CA Workload Automation EE (xtype: e), specify the encryption key defined by ENCRYPT KEYNAME(keyname) in the CA Workload Automation EE AGENTDEF data set. You must prefix the hexadecimal identifier 0x to this value, as shown in the following example:

```
xkey_to_manager: 0x0123456789ABCDEF0123456789ABCDEF
```

## xmachine Attribute—Define Connection Information for an External Instance

The xmachine attribute defines connection information for an external instance.

### Supported JIL Subcommands

This attribute is required for the [insert\\_xinst subcommand](#) (see page 856).

This attribute is optional for the [update\\_xinst subcommand](#) (see page 858).

### Syntax

This attribute has the following format:

`xmachine: ext_instance_node_names`

**`ext_instance_node_names`**

Defines the external instance's node names to be used for communication as follows:

- If the xtype attribute is set to **a** (an external CA Workload Automation AE instance), specify the application server machine name. To specify multiple application servers, separate each machine name with a comma.
- If the xtype attribute is set to **c** (a scheduling manager with CA AutoSys WA Connect Option) or **u** (a scheduling manager with CA UJMA), specify the machine name.
- If the xtype attribute value is **e** (CA Workload Automation EE), specify the host name of CA Workload Automation EE.

**Limits:** Up to 512 characters

### Example: Specify a Machine Name for an External Instance

This example defines an external CA Workload Automation AE instance named ACE. The instance's machine name is machineB. The instance communicates using port 9100 and default encryption.

```
insert_xinst: ACE
xtype: a
xmachine: machineB
xport: 9100
xcrypt_type: DEFAULT
```

## xmanager Attribute—Specify the Alias Name of a CA Workload Automation EE Instance

The xmanager attribute specifies the alias name of the CA Workload Automation EE instance.

**Note:** This attribute only applies to CA Workload Automation EE instances. To use this attribute, you must specify e for the xtype attribute.

### Supported JIL Subcommands

This attribute is required for the [insert\\_xinst](#) (see page 856) subcommand if the xtype attribute is set to e.

This attribute is optional for the [update\\_xinst](#) (see page 858) subcommand.

### Syntax

This attribute has the following format:

xmanager: *manager\_name*

*manager\_name*

Specifies the alias name of the CA Workload Automation EE instance.

**Limits:** Up to 16 characters

**Note:** You cannot use the same alias name for multiple CA Workload Automation EE instances. If CA Workload Automation AE detects another instance with the same alias name, you will get an error.

## xport Attribute—Specify the Port Number of the External Instance

The xport attribute specifies the port number of the external instance.

**Note:** This attribute only applies to CA Workload Automation AE and CA Workload Automation EE instances. To use this attribute, you must specify a or e for the xtype attribute.

### Supported JIL Subcommands

This attribute is required for the [insert\\_xinst](#) (see page 856) subcommand if the xtype attribute is set to a or e.

This attribute is optional for [update\\_xinst](#) (see page 858) subcommand.

### Syntax

This attribute has the following format:

**xport: *port***

***port***

Specifies the port number of the external instance.

**Limits:** 1-65535

## xtype Attribute—Specify the External Instance Type

The xtype attribute specifies the type of external instance.

### Supported JIL Subcommands

This attribute is required for the [insert\\_xinst subcommand](#) (see page 856).

### Syntax

This attribute has the following format:

`xtype: ext_instance_type`

#### ***ext\_instance\_type***

Specifies the external instance type. Options in the following:

- a—Identifies a remote CA Workload Automation AE application server instance.
- c—Identifies a CA AutoSys Workload Automation Connect Option instance.
- u—Identifies a CA Universal Job Management Agent or CA NSM instance.
- e—Identifies a CA Workload Automation EE instance.

### Example: Define an External CA Workload Automation AE Instance

This example defines an external CA Workload Automation AE application server instance named ACE to run on machineA and port 9000 using default encryption:

```
insert_xinst: ACE
xtype: a
xmachine: machineA
xport: 9000
```



# Chapter 9: JIL Monitor and Report Definitions

---

This chapter describes the JIL subcommands and attributes that you can use to monitor and generate reports about CA Workload Automation AE jobs. To define, update, or delete a monitor or report, you specify the appropriate JIL subcommand and attribute statements to the `jil` command.

## **delete\_monbro Subcommand—Delete a Monitor or Report**

The `delete_monbro` subcommand deletes the specified monitor or report from the CA Workload Automation AE database.

### **Syntax**

This subcommand has the following format:

`delete_monbro: monbro_name`

***monbro\_name***

Identifies an existing monitor or report to delete from the CA Workload Automation AE database.

**Limits:** Up to 30 alphanumeric characters

### **Example: Delete a Monitor**

This example deletes the monitor `track_alarm`:

```
delete_monbro: track_alarm
```

## insert\_monbro Subcommand—Define a Monitor or Report

The insert\_monbro subcommand defines a new monitor or report. Monitors are used to monitor CA Workload Automation AE events and to watch for specific occurrences, such as alarms. Reports are used to filter and report CA Workload Automation AE events.

To use a monitor or report, you must define it and run it. Defining it alone has no effect.

### Syntax

This subcommand has the following format:

`insert_monbro: monbro_name`

#### ***monbro\_name***

Defines a unique name for the monitor or report to define.

**Limits:** Up to 30 alphanumeric characters; the value can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-); the value cannot contain embedded blanks or tabs

### Required Attributes

To define a monitor or report, you must specify the following attribute following the insert\_monbro subcommand:

- [mode](#) (see page 881)

### Optional Attributes

You can specify the following optional attributes when defining a monitor or report:

- [alarm](#) (see page 873)
- [all\\_events](#) (see page 875)
- [all\\_status](#) (see page 876)
- [failure](#) (see page 878)
- [job\\_filter](#) (see page 879)
- [job\\_name](#) (see page 880) (required to monitor or report a single box or job)
- [restart](#) (see page 882)
- [running](#) (see page 883)
- [starting](#) (see page 884)
- [success](#) (see page 885)
- [terminated](#) (see page 886)

If you are defining a monitor, you can specify the following additional optional attribute:

- [alarm\\_verif](#) (see page 874)

If you are defining a report, you can specify the following additional optional attributes:

- [after\\_time](#) (see page 871)
- [currun](#) (see page 877)

#### **Example: Define a Report**

This example defines the report success\_report that browses all jobs for success in the current or most recent run of the job:

```
insert_monbro: success_report
mode: b /* "browser" can also be specified */
success: y
job_filter: a /* the default */
currun: y /* the default */
```

#### **Example: Define a Monitor**

This example defines the monitor alarm\_monitor to watch for alarms on all jobs.

```
insert_monbro: alarm_monitor
mode: m /* "monitor" can also be specified */
alarm: y
job_filter: a /* the default */
```

## update\_monbro Subcommand—Update a Monitor or Report

The update\_monbro subcommand updates an existing monitor or report.

Attributes in the existing monitor or report definition that are not explicitly replaced by attributes in the update\_monbro input retain their original settings after the update. When you need to reset many attributes, it may be more efficient to delete and redefine the monitor or report.

### Syntax

This subcommand has the following format:

update\_monbro: *monbro\_name*

***monbro\_name***

Identifies an existing monitor or report to update in the CA Workload Automation AE database.

**Limits:** Up to 30 alphanumeric characters

### Optional Attributes

You can specify the following optional attributes when updating a monitor or report:

- [alarm](#) (see page 873)
- [all\\_events](#) (see page 875)
- [all\\_status](#) (see page 876)
- [failure](#) (see page 878)
- [job\\_filter](#) (see page 879)
- [job\\_name](#) (see page 880)
- [mode](#) (see page 881)
- [restart](#) (see page 882)
- [running](#) (see page 883)
- [starting](#) (see page 884)
- [success](#) (see page 885)
- [terminated](#) (see page 886)

If you are updating a monitor, you can specify the following additional optional attribute:

- [alarm\\_verif](#) (see page 874)

If you are updating a report, you can specify the following additional optional attributes:

- [after\\_time](#) (see page 871)
- [currun](#) (see page 877)

#### **Example: Update a Monitor**

This example changes the monitor alarm\_monitor, so that it tracks all occurrences of jobs changing to TERMINATED status in addition to the alarms already specified in the monitor definition:

```
update_monbro: alarm_monitor
terminated: y
```

## **after\_time Attribute—Specify the Report Start Date and Time**

The after\_time report attribute specifies the start date and time for the report you are defining. Only events that occur after the specified date and time are reported.

#### **Supported JIL Subcommands**

This attribute is optional for the following subcommands (report definitions only):

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

## Syntax

This attribute has the following format:

`after_time: "date_time"`

### ***date\_time***

Defines the reporting start date in the format "*MM/DD/[YY]YY hh:mm*", where *MM* is the month, *DD* is the day, [*YY*]*YY* is the year, *hh* is the hour (in 24-hour format), and *mm* is the minutes. You must enclose the *date\_time* value in quotation marks ("").

**Default:** When the `currun` attribute is set to **no**, the `after_time` attribute defaults to 12:00 a.m. on the specified day. When the `currun` attribute is set to **yes**, the `after_time` attribute is ignored.

When you omit the date, the `after_time` attribute defaults to the current day.

**Note:** When you enter a two-digit year, CA Workload Automation AE saves the setting to the database as a four-digit year. If you enter 79 or less, the product precedes your entry with 20; if you enter 80 or greater, the product precedes your entry with 19.

## Example: Specify the Start Data and Time for a Report

This example report all events that occur after 2:00 p.m. on October 1, 2005:

`after_time: "10/01/2005 14:00"`

## alarm Attribute—Specify Whether to Track Alarms

The alarm monitor/report attribute specifies whether to track alarms for the specified jobs in the monitor or report you are defining. You can track alarms and other events in the same monitor or report.

**Note:** When you define the all\_events attribute, all alarms are tracked, regardless of the alarm attribute setting.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

alarm: y | n

y

Tracks alarms in the monitor or report.

**Note:** You can specify 1 instead of y.

n

Does not track alarms in the monitor or report. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Alarms

This example sets the monitor or report to track alarms:

```
alarm: y
```

## alarm\_verif Attribute—Specify Whether the Monitor Requires an Operator Response to Alarms

The alarm\_verif monitor attribute specifies whether the monitor you are defining requires an operator response to alarm events. When the monitor detects an alarm event, it prompts the operator for a response and repeats the prompt every 20 seconds until the operator responds. When the operator responds (typically by entering a comment at the prompt), the monitor time stamps the response and records it in the database along with the alarm event.

### Supported JIL Subcommands

This attribute is optional for the following subcommands (monitor definitions only):

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

alarm\_verif: y | n

y

Issues alarms until an operator responds.

**Note:** You can specify 1 instead of y.

n

Specifies that the monitor does not require operator verification of alarms. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor to Require Operator Verification of Alarms

This example sets the monitor to require operator verification of alarms:

```
alarm_verif: y
```

## all\_events Attribute—Specify Whether to Track all Events for a Job

The all\_events monitor/report attribute specifies whether to track all events for the specified jobs in the monitor or report you are defining. This attribute specifies whether any event filtering is in effect.

When you set this attribute to **y** or **1**, the product ignores other event filtering attributes and reports all events (including job status events, alarms, and manually generated events such as when you start a job), regardless of source, for the specified jobs.

When you set this attribute to **n** or **0**, the product uses other event filtering attributes (such as alarm, currun, failure, and so on) to select the events to track for the selected jobs.

**Note:** To monitor all events for all jobs, use the following command to display the scheduler log in real time instead of running a monitor:

```
autosyslog -e
```

Running a monitor adds another connection to the database and establishes a process that continually polls the database. This can significantly impact system performance. Using the scheduler log to monitor all jobs is more efficient and provides more diagnostic information than a monitor.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

all\_events: **y** | **n**

**y**

Tracks all events for the specified jobs.

**Note:** You can specify 1 instead of y.

**n**

Does not track all events for the specified jobs. This is the default.

**Note:** You can specify 0 instead of n.

#### **Example: Set the Monitor or Report to Track all Events**

This example sets the monitor or report to track all events for the specified jobs:

```
all_events: y
```

## **all\_status Attribute—Specify Whether to Track all Job Status Events**

The all\_status monitor/report attribute specifies whether to track all job status events for the specified jobs in the monitor or report you are defining. Job status events occur whenever a job's status changes.

When the all\_events attribute is set to **y** or **1**, the product ignores the all\_status attribute.

#### **Supported JIL Subcommands**

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

#### **Syntax**

This attribute has the following format:

all\_status: y | n

**y**

Tracks all job status events for the specified jobs.

**Note:** You can specify 1 instead of y.

**n**

Does not track all job status events for the specified jobs. This is the default.

**Note:** You can specify 0 instead of n.

#### **Example: Set the Monitor or Report to Track all Job Status Events**

This example sets the monitor or report to track all job status events:

```
all_status: y
```

## currun Attribute—Specify Whether to Report Events in the Current Run

The currun report attribute specifies whether to report only the events in the current or most recent run of the specified jobs in the report you are defining.

When you set the currun attribute to **n** or **0**, you must define the after\_time attribute.

### Supported JIL Subcommands

This attribute is optional for the following subcommands (report definitions only):

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

currun: **y** | **n**

**y**

Reports events only for the current or most recent job run. This is the default.

**Note:** You can specify 1 instead of y.

**n**

Does not restrict reporting to events for the current or most recent job run.

**Note:** You can specify 0 instead of n.

### Example: Report Only Events in the Current Run

This example reports only events in the current or most recent run of the specified jobs:

currun: **y**

## failure Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to FAILURE

The failure monitor/report attribute specifies whether to track the job status event generated when a job changes to the FAILURE status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

`failure: y | n`

**y**

Tracks the job status event generated when a job enters the FAILURE status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the FAILURE status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set Monitor or Report to Track Occurrences of Jobs Changing to FAILURE Status

This example sets the monitor or report to track occurrences of jobs changing to FAILURE status:

`failure: y`

## job\_filter Attribute—Specify the Jobs to Track

The job\_filter monitor/report attribute specifies which jobs to track in the monitor or report you are defining. The combination of the event filters and the job filter determines which events to track.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

job\_filter: **a** | **b** | **j**

**a**

Tracks all jobs (no job filtering). This is the default.

**Note:** You can specify **all** instead of **a**.

**b**

Tracks a single box and the jobs it contains. You must also define the job\_name attribute to specify the name of the job to track.

**Note:** You can specify **box** instead of **b**.

**j**

Tracks a single job. You must also define the job\_name attribute to specify the name of the job to track.

**Note:** You can specify **job** instead of **j**.

### Example: Set the Monitor or Report to Only Track Events Specified by job\_name Attribute

This example sets the monitor or report to only track events in the box specified by the job\_name attribute:

```
job_filter: b
```

## job\_name Attribute—Identify the Job for Which to Track Events

The job\_name monitor/report attribute identifies the box or job for which the monitor or report you are defining should track events. The combined event filters, job filter, and job name determines which events to track. The job\_name attribute is required when you set the job\_filter attribute to **j** (a single job) or **b** (a box and its jobs); otherwise, it is ignored.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

job\_name: *name*

*name*

Identifies the existing job or box job for which to track events.

**Limits:** Up to 64 alphanumeric characters; can include the following characters: a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-); cannot include embedded spaces or tabs

### Example: Track Events in a Box Job

This example only tracks events in the box job EOD\_Box (and the jobs it contains), set the job\_filter attribute to **b**:

job\_name: EOD\_Box

## mode Attribute—Specify Whether to Define a Monitor or Report

The mode monitor/report attribute specifies whether to define a monitor or report.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

`mode: monitor | browser`

#### **monitor**

Defines a monitor.

**Note:** You can specify **m** instead of monitor.

#### **browser**

Defines a report.

**Note:** You can specify **b** instead of browser.

### Example: Define a Report

This example defines a report:

`mode: b`

## restart Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to RESTART

The restart monitor/report attribute specifies whether to track the job status event generated when a job changes to the RESTART status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

`restart: y | n`

**y**

Tracks the job status event generated when a job enters the RESTART status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the RESTART status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Occurrences of Jobs Changing to RESTART Status

This example sets the monitor or report to track occurrences of jobs changing to RESTART status:

`restart: y`

## running Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to RUNNING

The running monitor/report attribute specifies whether to track the job status event generated when a job changes to the RUNNING status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

running: **y** | **n**

**y**

Tracks the job status event generated when a job enters the RUNNING status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the RUNNING status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Occurrences of Jobs Changing to RUNNING Status

This example sets the monitor or report to track occurrences of jobs changing to RUNNING status:

running: **y**

## starting Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to STARTING

The starting monitor/report attribute specifies whether to track the job status event generated when a job changes to the STARTING status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

`starting: y | n`

**y**

Tracks the job status event generated when a job enters the STARTING status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the STARTING status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Occurrences of Jobs Changing to STARTING Status

This example sets the monitor or report to track occurrences of jobs changing to STARTING status:

`starting: y`

## success Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to SUCCESS

The success monitor/report attribute specifies whether to track the job status event generated when a job changes to the SUCCESS status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

success: y | n

**y**

Tracks the job status event generated when a job enters the SUCCESS status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the SUCCESS status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Occurrences of Jobs Changing to SUCCESS Status

This example sets the monitor or report to track occurrences of jobs changing to SUCCESS status:

success: y

## terminated Attribute—Specify Whether to Track Job Events When a Jobs Status Changes to TERMINATED

The terminated monitor/report attribute specifies whether to track the job status event generated when a job changes to the TERMINATED status.

The product ignores this attribute when either the all\_events or the all\_status attribute is set to **y** or **1**.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_monbro](#) (see page 868)
- [update\\_monbro](#) (see page 870)

### Syntax

This attribute has the following format:

terminated: y | n

**y**

Tracks the job status event generated when a job enters the TERMINATED status.

**Note:** You can specify 1 instead of y.

**n**

Does not track the job status event generated when a job enters the TERMINATED status. This is the default.

**Note:** You can specify 0 instead of n.

### Example: Set the Monitor or Report to Track Occurrences of Jobs Changing to TERMINATED Status

This example sets the monitor or report to track occurrences of jobs changing to TERMINATED status:

terminated: y

# Chapter 10: JIL Blob and Glob Definitions

---

This chapter describes the JIL subcommands and attributes that you can use to insert and delete CA Workload Automation AE binary large objects tied to jobs (blobs) and global binary large objects (globs). To insert or delete a blob or glob, you specify the appropriate JIL subcommand and attribute statements to the jil command.

## **delete\_blob Subcommand—Delete a Blob from the Database**

The delete\_blob subcommand decouples the highest version of a blob from an existing job and deletes it from the database.

### **Syntax**

This subcommand has the following format:

`delete_blob: blob_name`

#### ***blob\_name***

Specifies the name of an existing job defined in the database to which the blob belongs.

**Limits:** Up to 64 characters

### **Required Attribute**

To delete a blob, you must specify the [blob\\_type attribute](#) (see page 893).

#### **Example: Delete a Blob from the Database**

This example deletes the highest version of input blob Blob1:

```
delete_blob: Blob1
blob_type: in
```

## delete\_glob Subcommand—Delete a Glob from the Database

The delete\_glob subcommand deletes a glob from the database.

### Syntax

This subcommand has the following format:

```
delete_glob: glob_name
glob_name
```

Specifies the name of a glob that is currently defined in the database.

**Limits:** Up to 64 characters

### Example: Delete a Glob from the Database

This example deletes the glob Glob1:

```
delete_glob: Glob1
```

## insert\_blob Subcommand—Add a Blob to the Database

The insert\_blob subcommand adds a new input blob definition tied to an existing job to the database.

### Syntax

This subcommand has the following format:

```
insert_blob: blob_name
blob_name
```

Specifies the name of an existing job defined in the database to which the blob belongs.

**Limits:** Up to 64 characters; the name can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-)

### Required Attribute

To add a blob, you must specify *one* of the following attributes:

- [blob\\_file](#) (see page 891)
- [blob\\_input](#) (see page 891)

#### **Example: Add a Blob with Input to the Database**

This example creates an input blob with textual input:

```
insert_blob: blob_input1
blob_input: <auto_blobt>Testing this blob</auto_blobt>
```

#### **Example: Add a Blob with a File to the Database**

This example creates an input blob with a file:

```
insert_blob: blob_input1
blob_file: /tmp/test.pl
```

## insert\_glob Subcommand—Add a Glob to the Database

The insert\_glob subcommand adds a new glob definition to the database.

### Syntax

This subcommand has the following format:

```
insert_glob: glob_name
glob_name
```

Defines a glob name.

**Limits:** Up to 64 characters; the name can only contain the following characters: a-z, A-Z, 0-9, period (.), underscore (\_), pound (#), and hyphen (-)

### Required Attributes

To add a glob, you must specify the following attributes:

- [blob\\_mode](#) (see page 892)
- [blob\\_file](#) (see page 891) or [blob\\_input](#) (see page 891)

#### Example: Add a Glob with Text Input to the Database

This example creates a glob with text input:

```
insert_glob: glob_input1
blob_mode: text
blob_input: <auto_blobt>Testing this glob</auto_blobt>
```

#### Example: Add a Glob with a Binary File to the Database

This example creates the glob with a binary file:

```
insert_glob: blob_input1
blob_mode: binary
blob_file: /tmp/test
```

## blob\_file Attribute—Specify File for the Blob

The blob\_file attribute specifies the file containing the data to be stored in the database.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_blob](#) (see page 888)
- [insert\\_glob](#) (see page 890)

**Note:** If you do not specify this attribute, you must specify the blob\_input attribute.

### Syntax

This attribute has the following format:

blob\_file: *file*  
*file*

Specifies the file to insert for the blob.

**Limits:** Up to 255 characters

## blob\_input Attribute—Specify Input for the Blob

The blob\_input attribute specifies one or more lines of text to be stored in the database. All text in the auto\_blobt XML-style metatags is stored literally and may span multiple lines.

### Supported JIL Subcommands

This attribute is optional for the following subcommands:

- [insert\\_blob](#) (see page 888)
- [insert\\_glob](#) (see page 890)

**Note:** If you do not specify this attribute, you must specify the blob\_file attribute.

### Syntax

This attribute has the following format:

blob\_input: <auto\_blobt>*input*</auto\_blobt>  
*input*

Specifies the text to insert for the blob.

## blob mode Attribute—Specify File Format for the Glob or Blob

The blob\_mode attribute specifies whether the file is a binary or a text file. The file is stored in the database as the specified format.

### Supported JIL Subcommands

This attribute is required for the [insert\\_glob subcommand](#) (see page 890).

This attribute is optional for the following subcommands:

- [insert\\_blob](#) (see page 888)
- [insert\\_job](#) (see page 289)

### Syntax

This attribute has the following format:

blob\_mode: b | t

**b**

Specifies that the file is a binary file.

**Note:** You can specify **binary** instead of b.

**t**

Specifies that the file is a text file.

**Note:** You can specify **text** instead of t.

## blob\_type Attribute—Specify File Type of the Job Blob

The blob\_type attribute specifies the type of a job blob.

### Supported JIL Subcommands

This attribute is required for the [delete\\_blob subcommand](#) (see page 887).

### Syntax

This attribute has the following format:

`blob_type: in | out`

**in**

Specifies that the latest version of the job input blob is deleted.

**Note:** You can specify `i` instead of `in`.

**out**

Specifies that the output or error blobs for the job are deleted.

**Note:** You can specify `o` instead of `out`.



# Chapter 11: System States

---

This chapter lists and describes the events, statuses, alarms, and exit codes that CA Workload Automation AE processes.

## Events

This section describes events that CA Workload Automation AE processes. Some events are generated internally, while some only occur when sent manually using the sendevent command.

The ujo\_event table uses the unique numeric codes in the following table to represent events:

| Numeric Code | Event Type      |
|--------------|-----------------|
| 101          | CHANGE_STATUS   |
| 105          | KILLJOB         |
| 106          | ALARM           |
| 107          | STARTJOB        |
| 108          | FORCE_STARTJOB  |
| 109          | STOP_DEMON      |
| 110          | JOB_ON_ICE      |
| 111          | JOB_OFF_ICE     |
| 112          | JOB_ON_HOLD     |
| 113          | JOB_OFF_HOLD    |
| 114          | CHK_MAX_ALARM   |
| 115          | HEARTBEAT       |
| 116          | CHECK_HEARTBEAT |
| 117          | COMMENT         |
| 118          | CHK_BOX_TERM    |
| 119          | DELETEJOB       |
| 120          | CHANGE_PRIORITY |
| 122          | CHK_RUN_WINDOW  |

| Numeric Code | Event Type          |
|--------------|---------------------|
| 125          | SET_GLOBAL          |
| 126          | SEND_SIGNAL         |
| 127          | EXTERNAL_DEPENDENCY |
| 129          | REFRESH_EXTINST     |
| 130          | MACH_OFFLINE        |
| 131          | MACH_ONLINE         |
| 133          | INVALIDATE_MACH     |
| 135          | CHK_START           |
| 136          | CHK_COMPLETE        |
| 137          | RELEASE_RESOURCE    |
| 138          | CHK_TERM_RUNTIME    |
| 140          | REPLY_RESPONSE      |
| 141          | STATE_CHANGE        |
| 142          | MACH_PROVISION      |
| 143          | RESTARTJOB          |

**More information:**

[sendevent Command—Send Events](#) (see page 183)

## ALARM (106)

The ALARM event is an informational event that invokes no action on its own. The ALARM event typically indicates the occurrence of one of the specific alarms. An alarm is typically an internal event, but you can also use this event to send alarms directly to operators.

**More information:**

[Alarms](#) (see page 909)

## ALERT (139)

The ALERT event is generated for jobs that monitor conditions continuously. Each time a specified condition occurs, an ALERT event is written to the scheduler log file. Alerts help you track and report each time that a monitored condition occurs. To stop a continuous monitor, you must complete the job manually by issuing the following command:

```
sendevent -E KILLJOB -J job_name
```

## CHANGE\_PRIORITY (120)

The CHANGE\_PRIORITY event modifies the job queue priority of the specified job to the priority specified in the -q parameter of the sendevent command.

Queue priority is the relative priority of all jobs in the queue. The lower the number, the higher the priority; **0** means to run the job immediately.

When the job is in QUE\_WAIT status, the CHANGE\_PRIORITY event changes the job priority immediately and may start the job. When the job is not yet in QUE\_WAIT status, the CHANGE\_PRIORITY event changes the priority only for the next run of the job. To permanently change a job's priority, you must edit the job definition.

## CHANGE\_STATUS (101)

The CHANGE\_STATUS event changes the specified job's status to the status defined in the -s parameter of the sendevent command. When the scheduler processes this event, it initiates any actions that are dependent upon the job's status.

You should not typically use the CHANGE\_STATUS event, because CA Workload Automation AE manages job state changes internally. To send this event, you must also define the -s *status* parameter in the sendevent command.

When you send a CHANGE\_STATUS event, you are changing the status of the job in the database; this event does not affect the current run of the job. That is, if you change the status to RUNNING, the status changes in the database but the job does not run. You must change the status to a termination state before the job can run again.

### More information:

[Status](#) (see page 906)

## CHECK\_HEARTBEAT (116)

The CHECK\_HEARTBEAT event instructs the scheduler to verify that all jobs for which heartbeat\_interval is set have generated heartbeats as scheduled. When a job fails to generate its scheduled heartbeat, the CHECK\_HEARTBEAT event generates a MISSING\_HEARTBEAT alarm. You can configure the scheduler to generate CHECK\_HEARTBEAT events automatically.

## CHK\_BOX\_TERM (118)

The internally generated CHK\_BOX\_TERM event instructs the scheduler to check whether a box job has run longer than the time specified in its max\_run\_alarm attribute.

## CHK\_MAX\_ALARM (114)

The internally generated CHK\_MAX\_ALARM event instructs the scheduler to check whether a job has run longer than the time specified in its max\_run\_alarm attribute.

## CHK\_RUN\_WINDOW (122)

The CHK\_RUN\_WINDOW event runs at the end of the interval specified in a job's run\_window attribute to verify that the job ran.

## CHK\_COMPLETE (136)

The CHK\_COMPLETE event is internally generated when the must\_complete\_times attribute is specified in a job definition. For each run of the job, one CHK\_COMPLETE event is generated for the upcoming must complete time. This event instructs the scheduler to check whether the job has completed by the specified time.

## CHK\_START (135)

The CHK\_START event is internally generated when the must\_start\_times attribute is specified in a job definition. For each run of the job, one CHK\_START event is generated for the upcoming must start time. This event instructs the scheduler to check whether the job has started by the specified time.

## CHK\_TERM\_RUNTIME (138)

The CHK\_TERM\_RUNTIME event is internally generated when a job is specified with the term\_run\_time attribute. If a job's run time exceeds the duration specified for the term\_run\_time attribute, the job terminates.

## COMMENT (117)

The COMMENT event is an informational event that attaches a message to the specified job. When used with the -J parameter of the sendevent command, the COMMENT event attaches the message to the job for a specific run. When used with the -C parameter of the sendevent command, the COMMENT event writes a comment directly to the scheduler log. The COMMENT event is a method for generating comments at run time and associating them with a specific job run.

## DELETEJOB (119)

The DELETEJOB event deletes the specified job from the database. If the job is a box, DELETEJOB deletes the box and all the jobs in it.

## EXTERNAL\_DEPENDENCY (127)

The EXTERNAL\_DEPENDENCY event is sent from an issuing instance to a different, receiving instance to signal that a cross-instance dependency was met.

## FORCE\_STARTJOB (108)

The manually generated FORCE\_STARTJOB event starts the specified job, regardless of whether its starting conditions are satisfied. Because you can use this command to start multiple instances of the same job, we recommend that you use the FORCE\_STARTJOB event only in extreme situations.

Do not force start a job that is in QUE\_WAIT state because jobs in QUE\_WAIT state are already “started” and are waiting for machine resources to become available to run. If you want a job in QUE\_WAIT to start running immediately, use the CHANGE\_PRIORITY event to change the job queue priority for the job to 0.

If a job fails inside a box and you fix the problem manually, use FORCE\_STARTJOB to rerun the job.

**Note:** If you use FORCE\_STARTJOB for a job that has a status of ON\_ICE or ON\_HOLD, upon completion (either success or failure), the status/condition does *not* change back to the previous condition.

Using FORCE\_STARTJOB for a job scheduled for a machine that is offline has no effect.

When you force start a job, CA Workload Automation AE starts the job immediately on the machine specified in the job definition, regardless of the current load on the machine or the job\_load specified for the job.

## HEARTBEAT (115)

The internally generated HEARTBEAT event is sent from the agent posting a heartbeat for a specific job.

## INVALIDATE\_MACH (133)

The INVALIDATE\_MACH event is generated when a new machine definition is inserted or an existing machine definition is updated. This event invalidates the cache that the scheduler maintains internally. The scheduler reads the new or updated machine definition from the database to its cache.

## JOB\_OFF\_HOLD (113)

The JOB\_OFF\_HOLD event instructs the scheduler to remove the specified job from ON\_HOLD status. If the job's starting conditions are met, the job starts. This event also removes a job for which you defined the auto\_hold attribute from ON\_HOLD status.

## JOB\_OFF\_ICE (111)

The manually generated JOB\_OFF\_ICE event instructs the scheduler to remove the specified job from ON\_ICE status. Jobs that are taken off ice do not start until the next time their starting conditions are met. If the specified job is in a box with a RUNNING status, the product attempts to start it, conditions permitting.

## JOB\_ON\_HOLD (112)

The manually generated JOB\_ON\_HOLD event instructs the scheduler to place the specified job in ON\_HOLD status. Jobs that are on hold do not start, and downstream dependent jobs do not run. A box cannot successfully complete if a job in it is on hold. If the job's status is STARTING or RUNNING, the JOB\_ON\_HOLD event has no effect.

## JOB\_ON\_ICE (110)

The manually generated JOB\_ON\_ICE event instructs the scheduler to put the specified job in ON\_ICE status.

When you put a job on ice, it affects downstream jobs dependent upon it. For example, the starting conditions for jobs downstream from JobA, which has been put on ice, evaluates as shown in the following table:

| If the condition is this: | It evaluates this: |
|---------------------------|--------------------|
| success (JobA)            | TRUE               |
| failure (JobA)            | FALSE              |
| terminated (JobA)         | FALSE              |
| done (JobA)               | TRUE               |
| notrunning (JobA)         | TRUE               |
| exitcode                  | FALSE              |

**Note:** CA Workload Automation AE does not allow you to put a job whose status is RUNNING or STARTING on ice.

## KILLJOB (105)

The manually generated KILLJOB event instructs the scheduler to kill the specified job. The KILLJOB event is supported for the following job types:

- |         |           |          |
|---------|-----------|----------|
| ■ CMD   | ■ DBMON   | ■ SAPPM  |
| ■ OMCPU | ■ DBPROC  | ■ SAPEVT |
| ■ OMD   | ■ DBTRIG  | ■ SAPJC  |
| ■ OMEL  | ■ SQL     | ■ SAPDA  |
| ■ OMIP  | ■ JMSSUB  | ■ PS     |
| ■ OMP   | ■ JMXSUB  | ■ OASG   |
| ■ OMS   | ■ SAP R/3 | ■ OASET  |
| ■ OMTF  | ■ SAPBWIP | ■ OACOPY |
| ■ FW    | ■ SAPBWPC | ■ ZOS    |
| ■ FT    | ■ SAPBDC  | ■ ZOSM   |

The KILLJOB event is *not* supported for the following job types:

- |            |          |           |
|------------|----------|-----------|
| ■ FTP      | ■ HTTP   | ■ JMXMREM |
| ■ SCP      | ■ JMSPUB | ■ JMXMOP  |
| ■ JAVARMI  | ■ JMXMAG | ■ WBSVC   |
| ■ ENTYBEAN | ■ JMXMAS | ■ I5OS    |
| ■ SESSBEAN | ■ JMXMC  | ■ ZOSDST  |
| ■ POJO     |          |           |

If you issue a KILLJOB event to the unsupported job types, you will get an error.

Additionally, the action from the KILLJOB event depends on the job type.

### Command jobs on UNIX

The KILLJOB event kills the process that is currently running and all the processes that it has spawned (for example, the process group). KILLJOB does not kill orphan processes. It sends a signal to the process, waits five seconds, then sends a second signal if the process is still there.

The default kill signals to send (typically SIGINT and SIGKILL) are specified in the configuration file with the KillSignals parameter. This enables you to program commands that react intelligently to the KILLJOB event. For UNIX processes, you can use the -k parameter in the sendevent command to define specific signals or to override default signals.

### Command jobs on Windows

Windows does not support the concept of process groups. When you issue a KILLJOB event for a job that runs an executable (\*.exe), KILLJOB kills the process specified in the command definition. When you issue a KILLJOB event for a job that runs something other than an \*.exe (for example, \*.bat, \*.cmd, or \*.com), KILLJOB terminates only the CMD.EXE process that CA Workload Automation AE used to launch the job. The Job Status is set according to the return code of the killed CMD.EXE process and can be one of the following: SUCCESS, FAILURE, or TERMINATED. Processes launched by user applications or batch (\*.bat) files are not killed.

You can use the SEND\_SIGNAL event to circumvent this KILLJOB limitation. Your application must watch for a specified semaphore signal and react accordingly, and you can use the SEND\_SIGNAL event to implement this behavior.

### Box jobs

The KILLJOB event changes the specified job's status to TERMINATED. No more jobs in the box are started. KILLJOB events are also sent for running jobs in the box defined with job\_terminator value of Y. Other jobs that are already running run to completion.

### File Watcher jobs

The KILLJOB event kills the file watcher job and changes its status to TERMINATED.

## MACH\_OFFLINE (130)

The MACH\_OFFLINE event instructs the scheduler to put the machine specified by the -N parameter in the sendevent command offline and puts jobs scheduled to start on it in PEND\_MACH status. This event can be manually or automatically generated.

## MACH\_ONLINE (131)

The MACH\_ONLINE event instructs the scheduler to put the machine in the -N parameter of the sendevent command online, remove jobs scheduled to run on that machine from PEND\_MACH status, and starts those jobs.

## MACH\_PROVISION (142)

The MACH\_PROVISION event is generated when a machine definition is inserted or updated with the provision attribute set to y. When the machine needs to be provisioned, a request is sent to CA Spectrum Automation Manager to provision (install and activate) the CA Workload Automation AE agent on the machine.

## REFRESH\_EXTINST (129)

When changes are made to the external instance definitions, the scheduler re-reads the external instance definitions and generates the REFRESH\_EXTINST event.

## RELEASE\_RESOURCE (137)

The RELEASE\_RESOURCE event releases renewable resources held by a job. The resources are released back to the resource pool. A resource is considered to be “held” when a job that consumes the resource does not release it back to the resource pool. The RELEASE\_RESOURCE event only releases resources that are held permanently. The event does not release resources that are currently being used by jobs.

## REPLY\_RESPONSE (140)

The REPLY\_RESPONSE event is generated when you issue the following command to respond to a WAIT\_REPLY\_ALARM:

```
sendevent -J job_name -E reply_response
```

## RESTARTJOB (143)

You can use the sendevent command to manually send the RESTARTJOB event to restart a failed z/OS job.

RESTARTJOB events are recorded in the scheduler log file (event\_demon.\$AUTOSERV on UNIX and event\_demon.%AUTOSERV% on Windows). These events are also displayed when you create a report using the autorep -J -d command. The report includes the events that are generated during the most recent job runs.

## SET\_GLOBAL (125)

The SET\_GLOBAL event sets a global variable. The command sends this event with a high priority so the scheduler processes the variable before any jobs reference it at run time. To send this event, you must also define the -G parameter in the sendevent command.

## SEND\_SIGNAL (126)

The SEND\_SIGNAL event sends a signal to a running job.

**UNIX Note:** To send this event, you must also define the -k and -J parameters in the sendevent command.

**Windows Note:** You can use the SEND\_SIGNAL event to signal a specific named semaphore. Then, you can program your application to watch for and react to that semaphore. To send this event, you must also define the -k and -J parameters in the sendevent command.

## STARTJOB (107)

The STARTJOB event starts the specified job when its starting conditions are satisfied. The STARTJOB event considers all starting conditions, such as time and date conditions and dependencies on other jobs. You cannot send a STARTJOB event to a job that is in a box. Jobs in boxes inherit the starting conditions of the box; as a result, they start when the box starts. Using the sendevent command to send the STARTJOB event is the recommended way to start a job manually.

## STATE\_CHANGE (141)

Some of the new job types go through different state changes when they run. For example, a z/OS Regular job can go through state changes for each step that runs. The STATE\_CHANGE event is generated for each state that a job goes through.

STATE\_CHANGE events are recorded in the scheduler log file (event\_demon.\$AUTOSERV on UNIX and event\_demon.%AUTOSERV% on Windows). These events are also displayed when you create a report using the autorep -J -d command. The report includes the events that are generated during the most recent job runs.

**Note:** You cannot use the sendevent command to manually send the STATE\_CHANGE event.

## STOP\_DEMON (109)

The manually generated STOP\_DEMON event instructs the scheduler to shut down. The STOP\_DEMON event is the only way to stop the scheduler; it does not stop the database.

## Status

Every job has a status, or current state, associated with it. This section describes the states through which CA Workload Automation AE moves jobs as it processes events.

The ujo\_chase, ujo\_event, ujo\_job\_runs, ujo\_job\_status, and ujo\_restart tables use the event status codes in the following table to represent statuses:

| Numeric Codes | Event Status |
|---------------|--------------|
| 1             | RUNNING      |
| 3             | STARTING     |
| 4             | SUCCESS      |
| 5             | FAILURE      |
| 6             | TERMINATED   |
| 7             | ON_ICE       |
| 8             | INACTIVE     |
| 9             | ACTIVATED    |
| 10            | RESTART      |
| 11            | ON_HOLD      |
| 12            | QUE_WAIT     |
| 13            | WAIT_REPLY   |
| 14            | PEND_MACH    |
| 15            | RESWAIT      |

### ACTIVATED (9)

The ACTIVATED status indicates that the top-level box in which the job resides is currently in the RUNNING state. This status is internally controlled and does not have an associated event.

## FAILURE (5)

The FAILURE status indicates one of the following:

- For a command job, the command exited with a code greater than the maximum success value specified for the job.
- For a box job, the failure conditions for the box evaluated to true.

## INACTIVE (8)

The INACTIVE status indicates that the job is inactive; it has no status, by (or in) itself. For example, a newly created job which has not run is inactive.

## PEND\_MACH (14)

The PEND\_MACH state indicates that the job can logically start and has a priority other than 0, but the machine to which it is assigned is currently offline. When the machine comes back online and the required load units become available, CA Workload Automation AE starts the job. To remove a job from the PEND\_MACH status, enter:

```
sendevent -E MACH_ONLINE -N machine
machine
```

Identifies the machine on which the job is schedule to run.

## ON\_HOLD (11)

The ON\_HOLD status indicates that the job is on hold and will not run until it receives the JOB\_OFF\_HOLD event.

## ON\_ICE (7)

The ON\_ICE status indicates that the job is removed from all conditions and logic, but is still defined. Operationally, this is the equivalent of deactivating the job.

## QUE\_WAIT (12)

The QUE\_WAIT status indicates that the job can logically start and has a priority other than 0, but the machines on which it can start do not have enough available load units. When the required load units become available, CA Workload Automation AE starts the job. To remove a job from QUE\_WAIT status, enter:

```
sendevent -E CHANGE_PRIORITY -J job_name -q 0
```

*job\_name*

Identifies the job for which to change the job queue priority to 0 (run immediately).

## RESTART (10)

The RESTART status indicates that a job that could not start because of hardware or application problems has been scheduled to restart.

## RESWAIT (15)

The RESWAIT status indicates that the job is waiting for a resource before it can continue running. When the resource is available, CA Workload Automation AE starts the job.

## RUNNING (1)

The RUNNING status indicates that the job is running. If the job is a box, the RUNNING status indicates that jobs in the box may be started (other conditions permitting). If the job is a command or file watcher job, the RUNNING status indicates that the specified process is running on the client.

## STARTING (3)

The STARTING status indicates that the scheduler has initiated the start procedure with the agent and the job is in the process of “coming up.” This status does not apply to box jobs.

## SUCCESS (4)

The SUCCESS status indicates that the job exited and is considered successful, as determined by the exit code (for a command job) or the success conditions (for a box job).

## TERMINATED (6)

The TERMINATED status indicates that the job was terminated.

## WAIT\_REPLY (13)

The WAIT\_REPLY status indicates that the job is waiting for a user reply before it can continue running.

# Alarms

This section lists the alarms that can be generated during job processing.

The ujo\_alarm and ujo\_event tables use the unique numeric codes in the following table to represent alarms:

| Numeric Code | Associated Alarm  |
|--------------|-------------------|
| 501          | FORKFAIL          |
| 502          | MINRUNALARM       |
| 503          | JOBFAILURE        |
| 505          | MAX_RETRY         |
| 506          | STARTJOBFAIL      |
| 507          | EVENT_HDLR_ERROR  |
| 508          | EVENT_QUE_ERROR   |
| 509          | JOBNOT_ONICEHOLD  |
| 510          | MAXRUNALARM       |
| 512          | RESOURCE          |
| 513          | MISSING_HEARTBEAT |
| 514          | CHASE             |
| 516          | DATABASE_COMM     |
| 517          | APP_SERVER_COMM   |
| 518          | VERSION_MISMATCH  |
| 519          | DB_ROLLOVER       |
| 520          | EP_ROLLOVER       |

| Numeric Code | Associated Alarm     |
|--------------|----------------------|
| 521          | EP_SHUTDOWN          |
| 522          | EP_HIGH_AVAIL        |
| 523          | DB_PROBLEM           |
| 524          | DUPLICATE_EVENT      |
| 525          | INSTANCE_UNAVAILABLE |
| 526          | AUTO_PING            |
| 529          | EXTERN_DEPS_ERROR    |
| 532          | MACHINE_UNAVAILABLE  |
| 533          | SERVICEDESK_FAILURE  |
| 534          | UNINOTIFY_FAILURE    |
| 535          | CPI_JOBNAME_INVALID  |
| 536          | CPI_UNAVAILABLE      |
| 537          | MUST_START_ALARM     |
| 538          | MUST_COMPLETE_ALARM  |
| 539          | WAIT_REPLY_ALARM     |
| 540          | KILLJOBFAIL          |
| 541          | SENDSIGFAIL          |
| 542          | REPLY_RESPONSE_FAIL  |
| 543          | RETURN_RESOURCE_FAIL |
| 544          | RESTARTJOBFAIL       |

## AUTO\_PING (526)

The AUTO\_PING alarm indicates that the autoping -M -A command cannot connect to the client. The alarm includes the name of the client.

## CHASE (514)

The CHASE alarm indicates that the chase command has found a problem with a job that is supposedly running. The alarm includes the job name and the problem encountered.

## CPI\_JOBNAME\_INVALID (535)

The CPI\_JOBNAME\_INVALID alarm indicates that the cross-platform interface has received a jobname that is not valid for the agent it is submitting the job to.

## CPI\_UNAVAILABLE (536)

The CPI\_UNAVAILABLE alarm indicates that the Cross Platform Interface is not running.

## DATABASE\_COMM (516)

The DATABASE\_COMM alarm indicates that the agent had trouble sending an event to the database. In most cases, the job ran successfully. Inspect the agent log file to determine the cause of the alarm.

## DB\_PROBLEM (523)

The DB\_PROBLEM alarm indicates that there is a problem with one of the databases. For example, this alarm can indicate a lack of free space. This alarm can trigger a user-specified notification procedure.

## DB\_ROLLOVER (519)

The DB\_ROLLOVER alarm indicates that CA Workload Automation AE has rolled over from dual event server to single event server mode. This alarm can trigger a user-specified notification procedure.

## DUPLICATE\_EVENT (524)

The DUPLICATE\_EVENT alarm indicates that the event server received duplicate events. Typically, this means that two schedulers are running, but this error may also result from event server configuration errors.

## EP\_HIGH\_AVAIL (522)

The EP\_HIGH\_AVAIL alarm can indicate any of the following:

- The third machine for resolving contentions between two schedulers cannot be reached
- The scheduler is shutting down
- There are other scheduler takeover problems

This alarm can trigger a user-specified notification procedure.

## EP\_ROLLOVER (520)

The EP\_ROLLOVER alarm indicates that the shadow scheduler is taking over processing.  
This alarm can trigger a user-specified notification procedure.

## EP\_SHUTDOWN (521)

The EP\_SHUTDOWN alarm indicates that the scheduler is shutting down. This may indicate a normal shutdown (for example, SEND\_EVENT triggered by sendevent -E STOP\_DEMON) or an error condition. This alarm can trigger a user-specified notification procedure.

## EVENT\_HDLR\_ERROR (507)

The EVENT\_HDLR\_ERROR alarm indicates that an error occurred in the scheduler while processing an event. Inspect the job associated with the event to verify whether manual intervention is required.

## EVENT\_QUE\_ERROR (508)

The EVEN\_QUE\_ERROR alarm indicates that an event could not be marked as processed. This is typically due to a problem with the event server. Contact CA Customer Support.

## EXTERN\_DEPS\_ERROR (529)

The EXTERN\_DEPS\_ERROR indicates that the cross-platform interface cannot send external dependencies to the remote node. The alarm identifies the failing remote node.

## FORKFAIL (501)

The FORKFAIL alarm indicates that the agent could not start the user command because it could not get a process slot on the UNIX machine. When this happens, CA Workload Automation AE automatically attempts a RESTART. This alarm can occur only when a job is running on a UNIX machine.

## KILLJOBFAIL (540)

The KILLJOBFAIL alarm indicates that the job is not killed when a KILLJOB event is issued.

## INSTANCE\_UNAVAILABLE (525)

The INSTANCE\_UNAVAILABLE alarm indicates that the event server of a receiving instance could not be reached when an attempt was made to communicate with it. The event server is probably down.

## JOBFAILURE (503)

The JOBFAILURE alarm indicates that a job failed or was terminated and its status is now FAILURE or TERMINATED.

## JOBNOT\_ONICEHOLD (509)

The JOBNOT\_ONICEHOLD alarm alerts users when a JOB\_ON\_HOLD or JOB\_ON\_ICE event could not place a job in ON\_HOLD or ON\_ICE status (for example, when the job is already running).

## MACHINE\_UNAVAILABLE (532)

The MACHINE\_UNAVAILABLE alarm indicates that a machine the scheduler communicated in the past is not responding to ping requests and will be taken OFFLINE.

## MAX\_RETRY (505)

The MAX\_RETRY alarm indicates that CA Workload Automation AE has reached the maximum number of restart attempts specified in the n\_retrys attribute without successfully starting the job. When the problem that resulted in the restart attempts is fixed, you must start the job manually.

## **MAXRUNALARM (510)**

The MAXRUNALARM indicates that a job has run longer than the interval defined in the job's max\_run\_alarm attribute. This alarm does not stop the job.

## **MINRUNALARM (502)**

The MINRUNALARM alarm indicates that a job finished in less time than was defined in the job's min\_run\_alarm attribute.

## **MISSING\_HEARTBEAT (513)**

The MISSING\_HEARTBEAT alarm indicates that a job has not generated a HEARTBEAT event during the interval specified in the job's heartbeat\_interval attribute. This alerts the operator of possible problems with the job.

## **MUST\_COMPLETE\_ALARM (538)**

The MUST\_COMPLETE\_ALARM indicates that a job has not completed by the time specified in the must\_complete\_times attribute.

## **MUST\_START\_ALARM (537)**

The MUST\_START\_ALARM indicates that a job has not started by the time specified in the must\_start\_times attribute.

## **REPLY\_RESPONSE\_FAIL (542)**

If a job is in WAIT\_REPLY state, it is waiting for a user reply before it can continue running. You can issue the following command to send a response:

```
sendevent -J job_name -E reply_response
```

If your response cannot be sent back to the job, the REPLY\_RESPONSE\_FAIL alarm is generated to indicate that the response failed to send.

## RESOURCE (512)

The RESOURCE alarm indicates that a resource needed for a job (for example, file space) is not available. The alarm text includes specific information about the problem. If CA Workload Automation AE encounters a resource problem, it attempts to restart the job after a suitable delay.

## RESTARTJOBFAIL (544)

The RESTARTJOBFAIL alarm indicates that a job could not be manually restarted using the sendevent -E RESTARTJOB command. The job could not be restarted because it is not in a FAILURE state or it is not a z/OS job.

RESTARTJOBFAIL alarms are displayed when you create a report using the autorep -J -d command.

## RETURN\_RESOURCE\_FAIL (543)

The RETURN\_RESOURCE\_FAIL alarm indicates that the resource manager is not able to return a resource.

## SENDSIGFAIL (541)

The SENDSIGFAIL alarm indicates that the SEND\_SIGNAL event was not sent to the job.

## SERVICEDESK\_FAILURE (533)

The SERVICEDESK\_FAILURE alarm indicates that the scheduler component was unable to open a CA Service Desk ticket for the failing job.

## STARTJOBFAIL (506)

The STARTJOBFAIL alarm indicates that CA Workload Automation AE was unable to start a job. This is typically caused by communication problems with the remote machine. CA Workload Automation AE attempts to restart the job.

## **UNINOTIFY\_FAILURE (534)**

The UNINOTIFY\_FAILURE alarm indicates that the scheduler component was unable to send a notification for the requesting job to the Notification Services component of CA NSM.

## **VERSION\_MISMATCH (518)**

The VERSION\_MISMATCH alarm is generated by the agent when the calling routine (for example scheduler, chase, clean\_files, autoping, and so on) is at a different version number than the agent. Inspect the agent log file for the exact version mismatch and install the proper agent version.

## **WAIT\_REPLY\_ALARM (539)**

The WAIT\_REPLY\_ALARM alarm indicates that a job is waiting for a user reply before it can continue running.

## Exit Codes

When you use the autosyslog -J command to display the agent log file for a job, it may contain an entry containing one of the exit codes listed in this section.

When the exit code contains two numbers in parentheses—for example (0 1)—the first number is the UNIX signal, and the second number is the exit code. When a job is killed or terminated the exit code remains at 0, which is what it was set to when the job started.

### **15 (15 0)**

The job was terminated by a UNIX kill –15.

### **101 (0 101)**

The job received a CHANGE\_STATUS event. For example, the job was changed to a TERMINATED or FAILURE status.

### **121 (0 121)**

The product could not open the file specified in the std\_in\_file attribute. The input file does not exist or is inaccessible. Check permissions.

### **122 (0 122)**

The product could not open the file specified in the std\_out\_file attribute. The output directory does not exist or is inaccessible. Check permissions and verify that the file system is not full.

### **123 (0 123)**

The product could not open the file specified in the std\_err\_file attribute. The output directory does not exist or is inaccessible. Check permissions and verify that the file system is not full.

### **127 (0 127)**

The directory that contains the executable is not in the command search PATH.

## **256 (0 1)**

The product could not execute the command. Possible causes include the following:

- The command or file to execute does not exist.
- The file to execute is not executable. Check permissions.
- The file to execute is in a directory that is not accessible. Check permissions (ls -ld). If the directory is NFS-mounted, verify that the mount exists.
- If you are using a job environment file, the PATH variable may be incorrect:
  - The file to execute may not be in a directory that is specified in PATH.
  - The PATH command may use an undefined variable (which translates as blank space, causing the PATH command to terminate before all the directories are defined to it).
  - The job environment file may use non-Bourne shell commands (for example, alias) and the expected settings (or aliases) may not exist.
  - The command contains incorrect options (for example, date –JASD).

## **512 (0 2)**

The command contains incorrect options (for example, awk 'junk').

### **-655 SYSTEM\_ERROR**

STARTJOB failures because the agent will not start.

### **-656 NO\_EXIT\_CODE**

exit\_code field in database is initialized to this.

### **-657 PROCESS\_MIA**

Set by a chase-generated FAILURE event (for example, chase cannot find the process).

# **Chapter 12: Database Tables and Views**

---

This chapter lists and describes the CA Workload Automation AE database tables and views.

Because CA Workload Automation AE uses a relational database, you can query the database to supply custom reports and information.

**Note:** Using SQL commands to change information in CA Workload Automation AE tables can cause your system to fail. The CA Workload Automation AE tables and views listed in the following sections are for documentation purposes only. They are not intended to be used as a public interface and are subject to change between releases or with maintenance.

By default, the CA Workload Automation AE OS user is "autosys". However, you can define a different ID for the OS user during installation.

## **ujo\_afm**

The ujo\_afm table stores AFM message sequence information from CA Workload Automation Agent, CA Workload Automation Agent for z/OS, and CA Workload Automation EE.

This table contains the following columns:

- afm\_num
- afm\_queue\_id
- events
- queue\_id
- sequence
- stamp

## ujointerfaces

The ujo\_afm\_strings table stores AFM message content details from CA Workload Automation Agent, CA Workload Automation Agent for z/OS, and CA Workload Automation EE.

This table contains the following columns:

- afm
- afm\_num
- afm\_queue\_id
- piece

## ujointerfaces

The ujo\_agent\_alias table stores the encryption key mapping for each alias used to communicate with CA Workload Automation Agent, CA Workload Automation Agent for z/OS, and CA Workload Automation EE.

This table contains the following columns:

- alias
- crypt\_key
- encryption\_type

## ujointerfaces

The ujo\_alamode table stores configuration information, such as autotrack and remote authentication levels, the number of event servers and timezone settings used by CA Workload Automation AE.

This table contains the following columns:

- type
- int\_val
- str\_val

## ujointerface

The ujointerface table stores all the alarms. Each alarm has a unique eoid (event object ID), which is the reference to the event that created the alarm.

The ujointerface table uses the alarm state codes in the following table to identify the alarm state:

| Numeric Code | Alarm State  |
|--------------|--------------|
| 43           | open         |
| 44           | acknowledged |
| 45           | closed       |

This table contains the following columns:

- |              |                 |
|--------------|-----------------|
| ■ eoid       | ■ state         |
| ■ alarm      | ■ the_user      |
| ■ alarm_time | ■ state_time    |
| ■ joid       | ■ event_comment |
| ■ evt_num    | ■ len           |
|              | ■ response      |

## ujointerface

The ujointerface table stores information regarding UUJMA computers with which the cross-platform interface has communicated.

This table contains the following columns:

- nod\_name
- nod\_ckpt
- nod\_boot

## ujointerface

The ujo\_asext\_config table stores the external instance definitions entered through the insert\_xinst command.

This table contains the following columns:

- |                                                                                                                                                      |                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>■ asext_autoserv</li><li>■ asext_instance_type</li><li>■ asext_nod_name</li><li>■ asext_manager_name</li></ul> | <ul style="list-style-type: none"><li>■ asext_crypt_type</li><li>■ asext_key</li><li>■ asext_port</li><li>■ asext_status</li></ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|

## ujointerface

The ujo\_audit\_info table stores most of the autotrack utility information. Use the -l parameter with the archive\_events command to archive this table. Together, the ujo\_audit\_info and ujo\_audit\_msg tables contain all of the autotrack utility information.

This table contains the following columns:

- audit\_info\_num
- entity
- time
- type

## ujointerface

The ujo\_audit\_msg table stores autotrack utility information additional to the information stored in the ujo\_audit\_info table. Use the -l parameter with the archive\_events command to archive this table. Together, the ujo\_audit\_info and ujo\_audit\_msg tables contain all of the autotrack utility information.

This table contains the following columns:

- audit\_info\_num
- seq\_no
- attribute
- value1
- value2
- is\_edit

## ujointerface

The ujo\_avg\_job\_runs table contains a row for each job which contains the job's calculated average run time based on the data in the ujo\_job\_runs table. The joid (job object ID) column is the unique key for each row. In addition, each row contains the num\_runs column, which indicates how many runs were used to calculate the average run time.

This table contains the following columns:

- joid
- avg\_runtime
- num\_runs

## ujointerface

The ujo\_calendar table contains a list of the dates for each calendar. You can define multiple calendars referenced by unique names.

This table contains the following columns:

- name
- cal\_id
- day

## ujointerface

The ujo\_chase table stores chase utility information. Information remains in the ujo\_chase table only temporarily.

This table contains the following columns:

- |            |               |
|------------|---------------|
| ■ nstart   | ■ status      |
| ■ joid     | ■ run_machine |
| ■ eoid     | ■ pid         |
| ■ job_type | ■ jc_pid      |
| ■ over_num | ■ job_ver     |
|            | ■ wf_joid     |

## ujointerface

The ujo\_chkpt\_rstart table stores event data when the cross-platform interface is unable to deliver it to the external manager. During the restart handshake, the cross-platform interface queries the ujo\_chkpt\_rstart table for event data to deliver to the external manager.

This table contains the following columns:

- |                |                  |
|----------------|------------------|
| ■ dest_machine | ■ ubc_compcod    |
| ■ dest_app     | ■ ubc_jobname    |
| ■ as_evt_time  | ■ ubc_setname    |
| ■ ubc_name     | ■ ubc_jobnumb    |
| ■ ubc_jobnumbr | ■ ubc_server     |
| ■ ubc_sysid    | ■ ubc_from_sysid |
| ■ ubc_date     | ■ enefill        |
| ■ ubc_time     | ■ ubt_cputime    |
| ■ ubc_procid   | ■ ubt_errcod     |
| ■ ubc_userid   |                  |

## ujointerface

The ujointerface table stores the communication details for the most recent AFM message to be received from a source computer.

This table contains the following columns:

- |                  |                 |
|------------------|-----------------|
| ■ acked_receive  | ■ seq_num       |
| ■ dest_node_name | ■ source        |
| ■ dest_queue_id  | ■ src_node_name |
| ■ product_id     | ■ src_queue_id  |

## ujointerface

The ujointerface table stores the communication details for the most recent AFM message to be sent to a destination computer.

This table contains the following columns:

- |                   |                 |
|-------------------|-----------------|
| ■ dest_host_alias | ■ seq_num       |
| ■ dest_node_name  | ■ src_node_name |
| ■ destination     | ■ src_queue_id  |
| ■ product_id      |                 |

## ujointerface

The ujo\_command\_job table stores Command (CMD) job definitions.

This table contains the following columns:

- |                                                                                                                                                                                          |                                                                                                                                                                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>■ chk_files</li><li>■ command</li><li>■ envvars</li><li>■ heartbeat_interval</li><li>■ interactive</li><li>■ is_script</li><li>■ job_ver</li></ul> | <ul style="list-style-type: none"><li>■ joid</li><li>■ over_num</li><li>■ shell</li><li>■ std_err_file</li><li>■ std_in_file</li><li>■ std_out_file</li><li>■ ulimit</li><li>■ userid</li></ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## ujointerface

The ujo\_cred table stores the user passwords entered through the autosys\_secure utility in encrypted format.

**Note:** If you use SQL commands to change this table, jobs will probably fail.

This table contains the following columns:

- cred\_domain
- domain\_name
- principal
- cred\_value
- owner

## ujointerface

The ujo\_cycle table stores cycle definitions.

This table contains the following columns:

- cycname
- cycperiod
- cycperst
- cycperen

## ujointerface

The ujo\_event table stores CA Workload Automation AE events. Each row in the table contains information about one event. During processing, the status changes are communicated using events and the corresponding states. The state is specified in the que\_status column. Each event has a unique identifier (eoid), which helps ensure that events cannot be lost, confused, or overwritten when you run with external instance job dependencies.

**Note:** If you use SQL commands to change this table, jobs will probably not run when scheduled or will remain in their current status indefinitely.

The table uses the following status codes in the que\_status column:

| Numeric Code | Associated Event                       |
|--------------|----------------------------------------|
| -3           | Unprocessed resorted event             |
| -4           | Resorted event waiting to be processed |
| -2           | Processing                             |
| -1           | Waiting to be processed                |
| 0            | Unprocessed                            |
| 2            | Processed                              |
| 3            | Processed with error                   |
| 4            | Unsent event                           |

This table contains the following columns:

- |                |                  |                    |
|----------------|------------------|--------------------|
| ■ eoid         | ■ alarm          | ■ ntry             |
| ■ joid         | ■ event_time_gmt | ■ text             |
| ■ job_name     | ■ exit_code      | ■ que_priority     |
| ■ box_name     | ■ pid            | ■ stamp            |
| ■ autoserv     | ■ jc_pid         | ■ evt_num          |
| ■ priority     | ■ run_num        | ■ que_status       |
| ■ event        | ■ box_joid       | ■ que_status_stamp |
| ■ status       | ■ job_ver        | ■ global_name      |
| ■ global_value | ■ proc_event_num | ■ mach_name        |
| ■ over_num     | ■ wf_name        | ■ que_status_proc  |
| ■ wf_joid      |                  |                    |

**More information:**

[Events](#) (see page 895)

## ujointerface

The ujointerface table stores duplicate events, which each have a unique eoid. Under normal conditions the ujointerface table is empty.

This table contains the following columns:

- |             |                  |                    |
|-------------|------------------|--------------------|
| ■ eoid      | ■ alarm          | ■ text             |
| ■ joid      | ■ event_time_gmt | ■ que_priority     |
| ■ autoserv  | ■ exit_code      | ■ stamp            |
| ■ priority  | ■ pid            | ■ evt_num          |
| ■ event     | ■ jc_pid         | ■ que_status       |
| ■ status    | ■ run_num        | ■ que_status_stamp |
| ■ mach_name | ■ ntry           | ■ counter          |

## ujointerfaceview

The ujointerfaceview view presents the information in the ujointerface table in a more readable form. Most notably, all the events, alarms, and statuses are displayed in an easily interpreted text format.

This view contains the following columns:

- |                |                  |                    |
|----------------|------------------|--------------------|
| ■ eoid         | ■ event          | ■ exit_code        |
| ■ stamp        | ■ eventtxt       | ■ que_status       |
| ■ evt_num      | ■ status         | ■ que_status_stamp |
| ■ joid         | ■ statustxt      | ■ run_num          |
| ■ job_name     | ■ alarm          | ■ ntry             |
| ■ box_name     | ■ alarmtxt       | ■ text             |
| ■ autoserv     | ■ event_time_gmt | ■ machine          |
| ■ job_ver      | ■ over_num       | ■ global_key       |
| ■ global_value | ■ wf_joid        |                    |

## ujointerface

The ujo\_ext\_calendar table stores extended calendar definitions.

This table contains the following columns:

- |             |              |
|-------------|--------------|
| ■ name      | ■ as_hact    |
| ■ cal_id    | ■ as_nwact   |
| ■ as_work   | ■ as_adj     |
| ■ as_holcal | ■ as_datecon |
| ■ as_cyccal |              |

## ujointerface

The ujo\_ext\_event table stores CA Workload Automation AE events when the scheduler is unable to deliver them to an external CA Workload Automation AE, CA AutoSys Workload Automation Connect Option, CA Universal Job Management Agent or CA NSM or CA Workload Automation EE instance. The ujo\_ext\_event table has most of the same columns as the ujo\_event table.

Each row in the table contains information about one event. During processing, the status changes are communicated using events and the corresponding states. The state is specified in the que\_status column. Each event has a unique identifier (eoid), which helps ensure that events cannot be lost, confused, or overwritten when you run with external instance job dependencies.

**Note:** If you use SQL commands to make changes, dependent jobs in external instances may not be scheduled.

The table uses the following status codes in the que\_status column:

| Numeric Code | Associated Event                              |
|--------------|-----------------------------------------------|
| 0            | Waiting to be sent to external instance       |
| 1            | Sent as part of external instance status ping |
| 2            | Send to external instance completed           |

This table contains the following columns:

- |                 |                  |                |
|-----------------|------------------|----------------|
| ■ eoid          | ■ alarm          | ■ ntry         |
| ■ joid          | ■ event_time_gmt | ■ text         |
| ■ job_name      | ■ exit_code      | ■ que_priority |
| ■ box_name      | ■ mach_name      | ■ evt_num      |
| ■ autoserv      | ■ pid            | ■ que_status   |
| ■ priority      | ■ jc_pid         | ■ global_value |
| ■ event         | ■ run_num        | ■ job_ver      |
| ■ status        | ■ box_joid       | ■ wf_name      |
| ■ global_name   | ■ over_num       |                |
| ■ req_object_id | ■ wf_joid        |                |

**More information:**

[Events](#) (see page 895)

## ujoint

The ujo\_ext\_job table stores the status of external job dependencies. If jobs on one instance depend on jobs that run on another instance, this table specifies the status of the referenced jobs running on the other instance.

This table contains the following columns:

- |                 |                   |
|-----------------|-------------------|
| ■ ext_job_name  | ■ run_num         |
| ■ ext_autoserv  | ■ ntry            |
| ■ status        | ■ exit_code       |
| ■ esp_status    | ■ esp_lstatus     |
| ■ status_time   | ■ ext_object_id   |
| ■ ext_wf_name   | ■ sched_from_time |
| ■ sched_to_time | ■ wf_name         |

## ujoint\_extended\_jobrun\_info

The ujo\_extended\_jobrun\_info table stores additional job run information such as workload object IDs and other job type specific information. Together, the ujo\_job\_run and ujo\_extended\_job\_run tables contain all of the run information for a job. Use the -j parameter with the archive\_events command to archive this table.

This table contains the following columns:

- |            |           |
|------------|-----------|
| ■ joid     | ■ run_num |
| ■ ntry     | ■ seq_num |
| ■ run_info | ■ type    |

## ujointerface

The ujo\_file\_watch\_job table stores File Watcher (FW) job definitions.

This table contains the following columns:

- change\_type
- change\_value
- file\_name
- file\_owner
- file\_size
- job\_ver
- joid
- over\_num
- poll
- recursive
- unc\_password
- unc\_userid
- user\_group
- watch\_type

## ujointerface

The ujo\_ftp\_job table stores FTP job definitions.

This table contains the following columns:

- compression
- direction
- job\_ver
- joid
- local\_file
- over\_num
- prelim\_cmd
- preserve\_time
- remote\_dir
- remote\_file
- scp\_protocol
- server\_name
- server\_port
- ssl
- target\_os
- transfer\_type
- userid

## ujointerface

The ujo\_glob table contains a row for each global variable. The glo\_name column is the unique key for each row.

This table contains the following columns:

- glo\_name
- value
- value\_set\_time
- owner
- permission

## ujointerface

The ujo\_globblob table stores global blobs.

This table contains the following columns:

- |               |                |
|---------------|----------------|
| ■ bname       | ■ create_stamp |
| ■ value       | ■ modify_user  |
| ■ create_user | ■ modify_stamp |

## ujointerface

The ujo\_ha\_process table stores details about the CA Workload Automation AE schedulers. It includes information used for high-availability processing such as the current process status and heartbeat.

This table contains the following columns:

- |              |                    |
|--------------|--------------------|
| ■ pid        | ■ hostname         |
| ■ time_stamp | ■ ha_designator_id |
| ■ port       | ■ ha_status_id     |
| ■ queue_id   |                    |

## ujointcodes

The ujo\_i5\_job table stores i5/OS job definitions.

This table contains the following columns:

- |                   |                    |
|-------------------|--------------------|
| ■ action          | ■ job_queue        |
| ■ cl_library_list | ■ job_ver          |
| ■ cur_library     | ■ joid             |
| ■ envvars         | ■ lda              |
| ■ exec_name       | ■ lib              |
| ■ exit_cc         | ■ name             |
| ■ job_description | ■ others           |
| ■ job_name        | ■ over_num         |
| ■ job_parms       | ■ process_priority |

## ujointcodes

The ujo\_intcodes table stores all the numeric codes—alarm, event, and status codes—used in other tables as well as their text representation. All CA Workload Automation AE processes reference the ujo\_intcodes table.

This table contains the following columns:

- fld
- code
- text

## ujointerface

The ujo\_job table stores the definitions common to all job types. The joid (job object ID) column is the unique key for each row. Together, the ujo\_job, ujo\_sched\_info, and the respective job type tables contain all of the job parameters.

**Note:** If you use SQL commands to change this table, jobs will probably not run when scheduled or will remain in their current status indefinitely.

This table contains the following columns:

- |                   |                  |                     |
|-------------------|------------------|---------------------|
| ■ joid            | ■ description    | ■ profile           |
| ■ job_name        | ■ box_terminator | ■ numero            |
| ■ job_type        | ■ job_terminator | ■ max_exit_success  |
| ■ box_joid        | ■ alert          | ■ send_notification |
| ■ owner           | ■ create_userid  | ■ service_desk      |
| ■ permission      | ■ has_blob       | ■ as_applic         |
| ■ n_retrys        | ■ has_condition  | ■ as_group          |
| ■ create_stamp    | ■ has_resource   | ■ destination_file  |
| ■ external_app    | ■ is_currver     | ■ has_box_failure   |
| ■ has_box_success | ■ over_num       | ■ has_notification  |
| ■ has_override    | ■ tag            | ■ has_service_desk  |
| ■ is_active       | ■ update_userid  | ■ job_class         |
| ■ job_qualifier   |                  | ■ job_ver           |
| ■ mach_name       |                  | ■ over_seed         |
| ■ sub_application |                  | ■ update_stamp      |
| ■ wf_joid         |                  |                     |

## ujoint

The ujo\_job\_cond table contains the breakdown of each job's condition statement. Each condition occupies a row in the table and is indexed so the product can reconstruct the entire job condition from a series of single, unique entries.

This table contains the following columns:

- |                     |                 |
|---------------------|-----------------|
| ■ cond_mode         | ■ operator      |
| ■ joid              | ■ value         |
| ■ indx              | ■ indx_ptr      |
| ■ type              | ■ test_glovalue |
| ■ cond_job_name     | ■ lookback_secs |
| ■ cond_job_autoserv | ■ over_num      |
|                     | ■ job_ver       |

## ujointerface

The ujo\_job\_resource\_dep table stores information about real and virtual resource dependencies.

This table contains the following columns:

- |            |            |
|------------|------------|
| ■ joid     | ■ amount   |
| ■ job_ver  | ■ free     |
| ■ roid     | ■ instance |
| ■ over_num | ■ value    |
| ■ wf_joid  | ■ op       |
| ■ restype  | ■ indx     |

## ujointerface

The ujo\_job\_runs table contains a row for each job run. The joid (job object ID), run\_num (run number), and ntry (number of tries to run the job) columns are the unique keys for each row. Each row in the table contains a job's start time, end time, run time (in seconds), completion status, and exit code. The scheduler updates this table. Together, the ujo\_job\_run and ujo\_extended\_job\_run tables contain all of the run information for a job. Use the -j parameter with the archive\_events command to archive this table.

This table contains the following columns:

- |             |                  |                     |
|-------------|------------------|---------------------|
| ■ joid      | ■ exit_code      | ■ reply_message     |
| ■ run_num   | ■ runtime        | ■ reply_response    |
| ■ ntry      | ■ evt_num        | ■ esp_status        |
| ■ starttime | ■ std_out_file   | ■ esp_lstatus       |
| ■ endtime   | ■ std_err_file   | ■ has_extended_info |
| ■ status    | ■ svcdesk_status | ■ job_ver           |
| ■ over_num  | ■ svcdesk_handle | ■ wf_joid           |
|             |                  | ■ run_machine       |

## ujointerface

The ujointerface table stores the current run information for every job. The joid column is the key field for each row. This table also contains information such as the current status, run number, last start time, last end time, and exit code for each job.

This table contains the following columns:

- |                  |                  |                        |
|------------------|------------------|------------------------|
| ■ joid           | ■ time_ok        | ■ over_num             |
| ■ status         | ■ exit_code      | ■ mvsflag              |
| ■ status_time    | ■ run_machine    | ■ mvsnode              |
| ■ run_num        | ■ que_name       | ■ mvsjobname           |
| ■ last_start     | ■ run_priority   | ■ qual                 |
| ■ last_end       | ■ next_priority  | ■ jno                  |
| ■ next_start     | ■ pid            | ■ entryno              |
| ■ run_window_end | ■ jc_pid         | ■ mvsautoserv          |
| ■ ntry           | ■ last_heartbeat | ■ has_extended_info    |
| ■ appl_ntry      | ■ evt_num        | ■ last_scheduled_start |
| ■ job_ver        | ■ jobset         |                        |
| ■ wf_joid        |                  |                        |

## ujointerface

The ujointerface table stores information about the relationships between parent and child jobs.

This table contains the following columns:

- depth
- joid
- lineage
- parent\_joid

## **ujoint.jobblob**

The ujoint.jobblob table contains job-specific blobs.

This table contains the following columns:

- joid
- type
- sequence
- data
- job\_ver
- over\_num

## ujoint

The ujo\_jobst view presents information from the ujo\_job, ujo\_job\_status, ujo\_sched\_info, ujo\_command\_job and ujo\_file\_watch\_job tables. At most two rows are returned for a job if the job has an active override.

This view contains the following columns:

|                    |                       |                    |
|--------------------|-----------------------|--------------------|
| ■ joid             | ■ job_terminator      | ■ status           |
| ■ job_name         | ■ std_in_file         | ■ status_time      |
| ■ job_type         | ■ std_out_file        | ■ run_num          |
| ■ box_joid         | ■ watch_file          | ■ ntry             |
| ■ owner            | ■ watch_file_min_size | ■ appl_ntry        |
| ■ permission       | ■ watch_interval      | ■ last_start       |
| ■ n_retrys         | ■ min_run_alarm       | ■ last_end         |
| ■ auto_hold        | ■ max_run_alarm       | ■ next_start       |
| ■ command          | ■ alarm_if_fail       | ■ exit_code        |
| ■ date_conditions  | ■ chk_files           | ■ run_machine      |
| ■ days_of_week     | ■ profile             | ■ que_name         |
| ■ run_calendar     | ■ heartbeat_interval  | ■ run_priority     |
| ■ exclude_calendar | ■ job_load            | ■ next_priority    |
| ■ start_times      | ■ priority            | ■ pid              |
| ■ start_mins       | ■ auto_delete         | ■ jc_pid           |
| ■ run_window       | ■ numero              | ■ time_ok          |
| ■ description      | ■ max_exit_success    | ■ last_heartbeat   |
| ■ term_run_time    | ■ has_blob            | ■ has_condition    |
| ■ box_terminator   | ■ has_override        | ■ has_service_desk |
| ■ box_name         | ■ job_ver             | ■ status_err_fill  |
| ■ has_notification | ■ wf_joid             |                    |
| ■ is_curver        |                       |                    |
| ■ time_zone        |                       |                    |

## ujointerface

The ujo\_jobtype table contains user-defined job type definitions and the attributes of jobs of the specified types.

This table contains the following columns:

- job\_type
- description
- exe\_name
- stage\_blob

## ujointerface

The ujo\_keymaster table stores all of the security and EDIT/EXEC superuser keys and the information associated with them.

**Note:** Do not use SQL commands to change information in this table or CA Workload Automation AE will not run. Further, do not use SQL commands to delete keys from this table unless instructed to do so by CA Customer Support.

This table contains the following columns:

- |            |            |
|------------|------------|
| ■ hostid   | ■ server   |
| ■ hostname | ■ dakey    |
| ■ product  | ■ not_used |
| ■ type     |            |

## ujointerface

The ujo\_last\_eoid\_counter table stores the number of the last event (that is, the last eoid) used by the scheduler.

This table contains the counter column.

## ujointerface

The ujointerface table stores details about the CA Workload Automation AE application servers.

This table contains the following columns:

- |                    |              |
|--------------------|--------------|
| ■ comm_alias       | ■ mgr_alias  |
| ■ hostname         | ■ pid        |
| ■ ma_designator_id | ■ port       |
| ■ ma_status_id     | ■ queue_id   |
| ■ comm_alias       | ■ time_stamp |

## ujointerface

The ujointerface table stores the machine definitions entered through the insert\_machine command.

This table contains the following columns:

- |                      |                       |
|----------------------|-----------------------|
| ■ parent_name        | ■ factor              |
| ■ que_name           | ■ max_load            |
| ■ type               | ■ agent_name          |
| ■ administrator      | ■ description         |
| ■ character_code     | ■ heartbeat_freq      |
| ■ heartbeat_attempts | ■ mach_status         |
| ■ mach_name          | ■ opsys               |
| ■ node_name          | ■ plugin_addtln_count |
| ■ plugin_list        | ■ port                |
| ■ prepjobid          | ■ provision           |

## ujointerface

The ujo\_meta\_enumerations table stores the set of possible JIL values for specific job type attributes. This table corresponds to the ujo\_meta\_properties, ujo\_meta\_types, and ujo\_meta\_rules tables.

This table contains the following columns:

- afm\_name
- enum\_name
- enumeration
- str\_value

## ujointerface

The ujo\_meta\_properties table stores the properties of JIL keywords. This table corresponds to the ujo\_meta\_enumerations, ujo\_meta\_types, and ujo\_meta\_rules tables.

This table contains the following columns:

- |               |                 |                |
|---------------|-----------------|----------------|
| ■ afm_name    | ■ default_value | ■ overrideable |
| ■ afm_subverb | ■ enum_name     | ■ parm_subtype |
| ■ afm_verb    | ■ internal      | ■ quoted       |
| ■ db_column   | ■ jil_name      | ■ report_seq   |
| ■ db_length   | ■ max_value     | ■ required     |
| ■ db_table    | ■ meta_id       | ■ rules        |
| ■ db_type     | ■ min_value     | ■ type_id      |

## ujointerface

The ujo\_meta\_rules table stores data that validates the JIL keywords that are dependent on another JIL keyword. This table corresponds to the ujo\_meta\_properties, ujo\_meta\_enumerations, and ujo\_meta\_types tables.

This table contains the following columns:

- data
- dep\_keyword
- keyword\_data
- rule\_name

## ujointerface

The ujo\_meta\_types table stores object types, such as valid job types, machines, and resources. This table corresponds to the ujo\_meta\_properties, ujo\_meta\_enumerations, and ujo\_meta\_rules tables.

This table contains the following columns:

- jil\_internal
- name
- object\_type
- seq\_num
- single\_occurrence
- sub\_type
- type\_id

## ujc\_micro\_focus\_job

The ujo\_micro\_focus\_job table stores Micro Focus (MICROFOCUS) job definitions.

This table contains the following columns:

- address\_type
  - envvars
  - jcl\_name
  - jcl\_type
  - jcl\_version
  - job\_ver
  - joid
  - mf\_userid
  - os\_userid
  - over\_num
  - server
  - server\_name

**ujointeractive**

The ujo\_monbro table contains a row for each monitor or report definition.

This table contains the following columns:

- name
  - mon\_mode
  - do\_output
  - sound
  - alarm\_verif
  - alarm
  - all\_events
  - all\_status
  - running
  - success
  - failure
  - terminate
  - starting
  - restart
  - on\_ice
  - on\_hold
  - job\_filter
  - job\_name
  - currun
  - after\_time
  - autoserv

## ujointerface

The ujo\_monitor\_object\_job table stores the definitions for the following jobs:

- CPU Monitoring job (OMCPU)
- Disk Monitoring job(OMD)
- IP Monitoring job (OMIP)
- Process Monitoring job (OMP)
- Text File Reading and Monitoring job (OMTF)
- Windows Service Monitoring job (OMS)

This table contains the following columns:

- |                |                 |                           |
|----------------|-----------------|---------------------------|
| ■ device       | ■ job_ver       | ■ search_first            |
| ■ encoding     | ■ joid          | ■ search_found            |
| ■ file_name    | ■ needed_status | ■ search_last             |
| ■ filter       | ■ no_change     | ■ search_type             |
| ■ inside_range | ■ over_num      | ■ size_format             |
| ■ ip_name      | ■ poll_interval | ■ text_file_filter_exists |
| ■ ip_port      | ■ process_name  | ■ time_format             |
|                |                 | ■ time_position           |
|                |                 | ■ use_avail               |

## ujointerface

The ujo\_monitor\_winevent\_log table stores Windows Event Log Monitoring (OMEL) job definitions.

This table contains the following columns:

- |                     |                  |
|---------------------|------------------|
| ■ computer_name     | ■ event_source   |
| ■ event_category    | ■ event_type     |
| ■ event_description | ■ from_date_time |
| ■ event_id          | ■ job_ver        |
| ■ event_log_name    | ■ joid           |
| ■ event_op          | ■ over_num       |

## ujos msg ack

The ujo\_msg\_ack table is used when the alarm\_verif attribute is set for a monitor. This table contains the alarm ID (eoid), identifies the user who responded to the alarm, the time the alarm was first generated, the time the alarm was acknowledged, and a brief operator comment.

This table contains the following columns:

- eoid
  - who
  - timein
  - timeack
  - comm

## ujointnextoid

The `ujoint_oid` table stores all of the oid (other ID) counters except the eoid used by the scheduler (which is stored in the `ujoint_Eoid_counter` table).

This table contains the following columns:

- oid
  - field

## ujointerfaces

The ujo\_oraapps table stores the definitions for the following jobs:

- Oracle E-Business Suite Copy Single Request (OACOPY)
- Oracle E-Business Suite Request Set (OASET)
- Oracle E-Business Suite Single Request (OASG)

This table contains the following columns:

- |                  |                       |                  |
|------------------|-----------------------|------------------|
| ■ appl_name      | ■ monitor_child       | ■ req_name       |
| ■ appl_name_type | ■ monitor_child_delay | ■ responsibility |
| ■ args           | ■ over_num            | ■ save_output    |
| ■ description    | ■ print_copies        | ■ steps          |
| ■ job_ver        | ■ print_style         | ■ use_defaults   |
| ■ joid           | ■ printer             | ■ userid         |

## ujointerfaces\_steps

The ujo\_oraapps\_steps table stores the values specified in the oracle\_programdata attribute of Oracle E-Business Suite Request Set (OASET) job definitions.

This table contains the following columns:

- |                |                 |
|----------------|-----------------|
| ■ args         | ■ print_style   |
| ■ job_ver      | ■ printer       |
| ■ joid         | ■ program_index |
| ■ over_num     | ■ save_output   |
| ■ print_copies | ■ step_num      |

## ujointerface

The ujo\_patch table stores information about patch installations.

This table contains the following columns:

- date\_applied
- description
- issue\_number

## ujointerface

The ujo\_peoplesoft\_job table stores Peoplesoft (PS) job definitions.

This table contains the following columns:

- |                   |                   |                   |
|-------------------|-------------------|-------------------|
| ■ arguments       | ■ email_webreport | ■ output_type     |
| ■ disable_restart | ■ envvars         | ■ over_num        |
| ■ dist_folder     | ■ job_name        | ■ process_name    |
| ■ dist_roles      | ■ job_ver         | ■ process_type    |
| ■ dist_users      | ■ joid            | ■ run_cntrl_args  |
| ■ email_addr_expd | ■ operator_id     | ■ run_cntrl_id    |
| ■ email_address   | ■ operator_type   | ■ run_cntrl_table |
| ■ email_log       | ■ output_format   | ■ server_name     |
| ■ email_subject   | ■ output_path     | ■ skip_parm_upd   |
| ■ email_text      |                   | ■ timezone        |

## ujointerface

The ujo\_proc\_event table stores all processed events.

The table uses the following status codes in the que\_status column:

| Numeric Code | Associated Event     |
|--------------|----------------------|
| 2            | Processed            |
| 3            | Processed with error |
| 4            | Unsent event         |

This table contains the following columns:

- eoid
- joid
- autoserv
- priority
- event
- status
- global\_value
- over\_num
- alarm
- event\_time\_gmt
- exit\_code
- mach\_name
- pid
- jc\_pid
- run\_num
- box\_joid
- job\_ver
- ntry
- text
- que\_priority
- stamp
- evt\_num
- que\_status
- que\_status\_stamp
- global\_name
- orig\_evt\_num
- wf\_joid

## ujointerface

The ujointerface table stores information about the supported real resource types.

This table contains the following columns:

- roid
- name
- keyword
- type
- optional
- max\_value
- min\_value

## ujointerface

The ujointerface table contains data used to generate a daily report of CA Workload Automation AE activity. Each day occupies a single row in the table.

This table contains the following columns:

- |                   |                  |                   |
|-------------------|------------------|-------------------|
| ■ hour            | ■ js_onice_p     | ■ jc_jedit_n      |
| ■ hour_display    | ■ js_inactive_n  | ■ jc_svcdesk_n    |
| ■ js_running_n    | ■ js_inactive_p  | ■ jc_svcdesk_p    |
| ■ js_running_p    | ■ js_activated_n | ■ ag_alarmres_n   |
| ■ js_starting_n   | ■ js_activated_p | ■ ac_alarm_n      |
| ■ js_starting_p   | ■ js_restart_n   | ■ ac_noresponse_n |
| ■ js_success_n    | ■ js_restart_p   | ■ ac_jobfail_n    |
| ■ js_success_p    | ■ jc_jrun_n      | ■ ac_stjobfail_n  |
| ■ js_failure_n    | ■ jc_jfail_n     | ■ ac_maxretry_n   |
| ■ js_failure_p    | ■ jc_jforce_n    | ■ ac_maxrunt_n    |
| ■ js_terminated_n | ■ jc_jrestart_n  | ■ ac_minrunt_n    |
| ■ js_terminated_p | ■ jc_quewait_n   | ■ ac_dbroll_n     |
| ■ js_onhold_n     | ■ jc_kill_n      | ■ ac_scroll_n     |
| ■ js_onhold_p     |                  | ■ ac_schshut_n    |
| ■ js_onice_n      |                  |                   |

## ujointerface

The ujo\_rep\_hourly table contains data used to generate an hourly report of CA Workload Automation AE activity. Each hour occupies a single row in the table.

This table contains the following columns:

- |                   |                  |                   |
|-------------------|------------------|-------------------|
| ■ hour            | ■ js_onice_p     | ■ jc_jedit_n      |
| ■ hour_display    | ■ js_inactive_n  | ■ jc_svcdesk_n    |
| ■ js_running_n    | ■ js_inactive_p  | ■ jc_svcdesk_p    |
| ■ js_running_p    | ■ js_activated_n | ■ ag_alarmres_n   |
| ■ js_starting_n   | ■ js_activated_p | ■ ac_alarm_n      |
| ■ js_starting_p   | ■ js_restart_n   | ■ ac_noresponse_n |
| ■ js_success_n    | ■ js_restart_p   | ■ ac_jobfail_n    |
| ■ js_success_p    | ■ jc_jrun_n      | ■ ac_stjobfail_n  |
| ■ js_failure_n    | ■ jc_jfail_n     | ■ ac_maxretry_n   |
| ■ js_failure_p    | ■ jc_jforce_n    | ■ ac_maxrunt_n    |
| ■ js_terminated_n | ■ jc_jrestart_n  | ■ ac_minrunt_n    |
| ■ js_terminated_p | ■ jc_quewait_n   | ■ ac_dbroll_n     |
| ■ js_onhold_n     | ■ jc_kill_n      | ■ ac_scroll_n     |
| ■ js_onhold_p     |                  | ■ ac_schshut_n    |
| ■ js_onice_n      |                  |                   |

## ujointerface

The ujointerface table contains data used to generate a monthly report of CA Workload Automation AE activity. Each month occupies a single row in the table.

This table contains the following columns:

- |                   |                  |                   |
|-------------------|------------------|-------------------|
| ■ hour            | ■ js_onice_p     | ■ jc_jedit_n      |
| ■ hour_display    | ■ js_inactive_n  | ■ jc_svcdesk_n    |
| ■ js_running_n    | ■ js_inactive_p  | ■ jc_svcdesk_p    |
| ■ js_running_p    | ■ js_activated_n | ■ ag_alarmres_n   |
| ■ js_starting_n   | ■ js_activated_p | ■ ac_alarm_n      |
| ■ js_starting_p   | ■ js_restart_n   | ■ ac_noresponse_n |
| ■ js_success_n    | ■ js_restart_p   | ■ ac_jobfail_n    |
| ■ js_success_p    | ■ jc_jrun_n      | ■ ac_stjobfail_n  |
| ■ js_failure_n    | ■ jc_jfail_n     | ■ ac_maxretry_n   |
| ■ js_failure_p    | ■ jc_jforce_n    | ■ ac_maxrunt_n    |
| ■ js_terminated_n | ■ jc_jrestart_n  | ■ ac_minrunt_n    |
| ■ js_terminated_p | ■ jc_quewait_n   | ■ ac_dbroll_n     |
| ■ js_onhold_n     | ■ jc_kill_n      | ■ ac_scroll_n     |
| ■ js_onhold_p     |                  | ■ ac_schshut_n    |
| ■ js_onice_n      |                  |                   |

## ujointerface\_tables

The ujointerface\_tables table contains data used to generate a weekly report of CA Workload Automation AE activity. Each week occupies a single row in the table.

This table contains the following columns:

- |                   |                  |                   |
|-------------------|------------------|-------------------|
| ■ hour            | ■ js_onice_p     | ■ jc_jedit_n      |
| ■ hour_display    | ■ js_inactive_n  | ■ jc_svcdesk_n    |
| ■ js_running_n    | ■ js_inactive_p  | ■ jc_svcdesk_p    |
| ■ js_running_p    | ■ js_activated_n | ■ ag_alarmres_n   |
| ■ js_starting_n   | ■ js_activated_p | ■ ac_alarm_n      |
| ■ js_starting_p   | ■ js_restart_n   | ■ ac_noresponse_n |
| ■ js_success_n    | ■ js_restart_p   | ■ ac_jobfail_n    |
| ■ js_success_p    | ■ jc_jrun_n      | ■ ac_stjobfail_n  |
| ■ js_failure_n    | ■ jc_jfail_n     | ■ ac_maxretry_n   |
| ■ js_failure_p    | ■ jc_jforce_n    | ■ ac_maxrunt_n    |
| ■ js_terminated_n | ■ jc_jrestart_n  | ■ ac_minrunt_n    |
| ■ js_terminated_p | ■ jc_quewait_n   | ■ ac_dbroll_n     |
| ■ js_onhold_n     | ■ jc_kill_n      | ■ ac_scroll_n     |
| ■ js_onhold_p     |                  | ■ ac_schshut_n    |
| ■ js_onice_n      |                  |                   |

## ujointerface

The ujointerface table stores the jobs on one instance that are referenced in the starting dependencies of jobs running on another instance.

This table contains the following columns:

- req\_pending\_delete
- req\_job\_name
- req\_autoserv
- req\_box\_name
- req\_workflow
- req\_remote\_node
- req\_object\_ID
- sched\_from\_time
- sched\_to\_time
- run\_num
- req\_trigger\_type

## ujointerface

The ujointerface table stores information about the jobs waiting on resources.

This table contains the following columns:

- joid
- wf\_joid
- run\_num
- ntry
- priority
- stamp

## ujointerface

The ujo\_sap\_arcspec table stores the archive information specified in the sap\_step\_parms of SAP job definitions.

This table contains the following columns:

- |                 |                |               |
|-----------------|----------------|---------------|
| ■ arc_client    | ■ arc_info     | ■ arc_storage |
| ■ arc_connect   | ■ arc_obj_type | ■ arc_text    |
| ■ arc_date      | ■ arc_path     | ■ arc_userid  |
| ■ arc_doc_class | ■ arc_printer  | ■ arc_version |
| ■ arc_doc_type  | ■ arc_protocol | ■ job_ver     |
| ■ arc_format    | ■ arc_report   | ■ joid        |
| ■ arc_host_link | ■ arc_service  | ■ over_num    |
|                 |                | ■ seq_num     |

## ujointerface

The ujo\_sap\_infodetail table stores the information specified in the sap\_ext\_table attribute of SAP BW InfoPackage (SAPBWIP) job definitions.

This table contains the following columns:

- |              |               |
|--------------|---------------|
| ■ field_name | ■ object_name |
| ■ high       | ■ operation   |
| ■ job_ver    | ■ over_num    |
| ■ joid       | ■ seq_num     |
| ■ low        | ■ sign        |

## ujointerface

The ujo\_sap\_job table stores SAP job definitions.

This table contains the following columns:

- abap\_name
- arc\_object
- arc\_variant
- arcspec
- chain\_id
- destination
- details
- error\_rate
- event\_id
- event\_parms
- extended\_log
- failure\_message
- info\_package
- istrigger
- job\_count
- job\_name
- job\_ver
- jobsteps
- joid
- logon\_client
- logon\_lang
- logon\_rfc
- logon\_targsys
- logon\_userid
- maillist
- monitor\_child
- over\_num
- process\_client
- process\_status
- process\_type
- process\_userid
- processed\_error\_rate
- prtspec
- recipients
- rfc
- sap\_job\_class
- save\_in\_outbox
- start\_mode
- step\_number
- strings
- success\_message
- target\_jobname
- web\_posting

## ujointerface

The ujo\_sap\_jobstep table stores the values specified in the sap\_step\_parms attributes of all SAP job definitions.

This table contains the following columns:

- abap\_lang
- abap\_name
- abap\_userid
- abap\_variant
- arcspec
- email\_address
- failure\_message
- job\_ver
- joid
- maillist
- over\_num
- prtspec
- seq\_num
- success\_message
- web\_posting

## uj0\_sap\_prtspec

The uj0\_sap\_prtspec table stores the printing information specified in the sap\_step\_parms attribute of all SAP job definitions.

This table contains the following columns:

- |               |                 |                    |
|---------------|-----------------|--------------------|
| ■ arch_mode   | ■ prt_dept      | ■ prt_priority     |
| ■ banner      | ■ prt_dest      | ■ prt_release      |
| ■ banner_page | ■ prt_dsn       | ■ prt_request_type |
| ■ expiration  | ■ prt_footer    | ■ prt_show_pwd     |
| ■ job_ver     | ■ prt_format    | ■ prt_text         |
| ■ joid        | ■ prt_hostpage  | ■ recipient        |
| ■ over_num    | ■ prt_immediate | ■ seq_num          |
| ■ prt_cols    | ■ prt_lines     | ■ spool_name       |
| ■ prt_copies  | ■ prt_new       |                    |

## ujointerface

The ujo\_sap\_recipient table stores the information specified in the sap\_recipients attribute of all SAP job definitions.

This table contains the following columns:

- blind\_copy
- copy
- express
- fax\_form
- form\_lang
- from\_city
- from\_company
- from\_dept
- from\_fax
- from\_faxext
- from\_name
- from\_street
- from\_tel
- from\_telex
- from\_title
- job\_ver
- joid
- noforward
- nowrap
- over\_num
- recipient
- recipient\_type
- reply
- send\_cover
- send\_now
- seq\_num
- to\_attn
- to\_city
- to\_country
- to\_dept
- to\_fax
- to\_name
- to\_street
- to\_title

## ujosched\_info

The ujo\_sched\_info table stores the scheduling information common to jobs of all types.

This table contains the following columns:

- alarm\_if\_fail
- exec\_time
- over\_num
- alert
- fail\_codes
- priority
- auto\_delete
- job\_load
- run\_calendar
- auto\_hold
- job\_ver
- run\_window
- continuous
- joid
- start\_mins
- continuous\_alert
- max\_run\_alarm
- start\_times
- date\_conditions
- min\_run\_alarm
- success\_codes
- days\_of\_week
- mode\_type
- term\_run\_time
- envvars
- must\_complete
- timezone
- exclude\_calendar
- must\_start

## ujoservice\_desk

The ujo\_service\_desk table stores information about Service Desk request attributes.

This table contains the following columns:

- attribute
- joid
- description
- over\_num
- impact\_level
- priority
- job\_ver
- severity

## ujosnmpjob

The ujosnmpjob table stores SNMP job definitions.

This table contains the following columns:

- |                 |              |                    |
|-----------------|--------------|--------------------|
| ■ auth_protocol | ■ mib        | ■ port             |
| ■ community     | ■ node       | ■ privacy_protocol |
| ■ host          | ■ over_num   | ■ snmp_userid      |
| ■ job_ver       | ■ params     | ■ version          |
| ■ joid          | ■ params_num |                    |

## ujosqljob

The ujosqljob table stores the definitions for the following jobs:

- Database Monitoring job (DBMON)
- Database Stored Procedure (DBPROC)
- Database Trigger job (DBTRIG)
- SQL job (SQL)

This table contains the following columns:

- |              |                     |                  |
|--------------|---------------------|------------------|
| ■ command    | ■ job_ver           | ■ params         |
| ■ criteria   | ■ joid              | ■ params_num     |
| ■ dbtype     | ■ monitor_condition | ■ rtn_type       |
| ■ dburl      | ■ monitor_type      | ■ trig_condition |
| ■ dbuserid   | ■ obj_name          | ■ trig_type      |
| ■ dbuserrole | ■ over_num          |                  |

## ujointerface.ujo\_sql\_proc\_parms

The ujo\_sql\_proc\_parms table stores the information specified in the sp\_arg attribute of Database Stored Procedure (DBPROC) job definitions.

This table contains the following columns:

- |                                                                                                                             |                                                                                                            |
|-----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>■ ignore</li><li>■ job_ver</li><li>■ joid</li><li>■ name</li><li>■ over_num</li></ul> | <ul style="list-style-type: none"><li>■ seq_num</li><li>■ sqltype</li><li>■ type</li><li>■ value</li></ul> |
|-----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|

## ujointerface.ujo\_strings

The ujo\_strings table stores string data.

This table contains the following columns:

- seq\_num
- str\_oid
- value

## ujointerface.ujo\_sys\_ha\_state

The ujo\_sys\_ha\_state table contains the current high-availability state for the system.

This table contains the ha\_state\_id column.

## ujointerface.ujo\_temp\_jobruns

The ujo\_temp\_jobruns table stores temporary job run information while the archive\_events command is running.

This table contains the following columns:

- joid
- run\_num

## ujointimezones

The ujo\_timezones table stores time zone information. The ujo\_timezones table also stores information about user-defined time zones created through the autotimezone command.

This table contains the following columns:

- name
- type
- zone

## ujouninotify

The ujo\_uninotify table stores the integration data for CA Service Desk and the Notification Services component of CA NSM. The Service Desk and Notification attributes for a job are stored in this table as a keyword/value pair.

This table contains the following columns:

- joid
- job\_ver
- notification\_id
- notification\_msg
- over\_num

## ujovirt\_resource

The ujo\_virt\_resource table stores all virtual resource definitions. During run time, this table is used to check for the resource amounts available.

This table contains the following columns:

- |                  |                |
|------------------|----------------|
| ■ roid           | ■ create_stamp |
| ■ mach_name      | ■ modify_user  |
| ■ type           | ■ modify_stamp |
| ■ amount_defined | ■ description  |
| ■ create_user    |                |

## **ujos\_virt\_resource\_lookup**

The `ujos_virt_resource_lookup` table stores the key for every virtual resource definition. The key is used to reference each virtual resource by name.

This table contains the following columns:

- roid
  - res\_name
  - global\_ind

## ujos\_virt\_resource\_status

The `ujr_virt_resource_status` table stores current status information about the virtual resources that are used by jobs. When a job is submitted, this table indicates the amount of resource the job uses. This table also keeps track of the renewable resources that are not freed by jobs.

This table contains the following columns:

- roid
  - mach\_name
  - joid
  - job\_ver
  - wf\_joid
  - run\_num
  - ntry
  - amount\_in\_use
  - time

## ujos\_wait\_QUE

The ujo\_wait\_que table stores information about jobs currently in QUE\_WAIT status. The information remains in the ujo\_wait\_que table temporarily.

This table contains the following columns:

- joid
  - job\_load
  - que\_name
  - max\_load
  - priority
  - stamp

## ujointerfaces

The ujo\_web\_services table stores Web Service (WBSVC) job definitions.

This table contains the following columns:

- authentication\_order
- bean\_name
- class\_name
- conn\_domain
- conn\_origin
- connection\_factory
- create\_params
- create\_params\_num
- credentials
- destination
- filter
- filter\_type
- finder\_name
- finder\_params
- finder\_params\_num
- global\_proxy\_defaults
- initial\_context\_factory
- j2ee\_userid
- job\_criteria
- job\_ver
- joid
- mbean
- message\_class
- method\_name
- modify\_params
- modify\_params\_num
- one\_way
- operation\_type
- over\_num
- params
- params\_num
- port\_name
- proxy\_domain
- proxy\_port
- proxy\_user
- servlet\_type
- use\_topic

## ujointerfaces

The ujo\_web\_services2 table stores details about Web Service (WBSVC) job definitions.

This table contains the following columns:

- |                |                     |                    |
|----------------|---------------------|--------------------|
| ■ job_ver      | ■ proxy_host        | ■ service_name     |
| ■ joid         | ■ proxy_origin_host | ■ success_pattern  |
| ■ operation    | ■ remote_name       | ■ target_namespace |
| ■ over_num     | ■ return_namespace  | ■ url              |
| ■ provider_url | ■ return_xml_name   | ■ wsdl_url         |

## ujointerfaces

The ujo\_wol\_job table stores Wake on LAN (WOL) job definitions.

This table contains the following columns:

- |             |            |
|-------------|------------|
| ■ broadcast | ■ over_num |
| ■ job_ver   | ■ ports    |
| ■ joid      | ■ wakepwd  |
| ■ mac       |            |

## ujointerface

The ujo\_workflow table stores information about workflows.

This table contains the following columns:

- |                 |                 |              |
|-----------------|-----------------|--------------|
| ■ create_time   | ■ generation    | ■ permission |
| ■ create_userid | ■ has_condition | ■ run_name   |
| ■ delete_time   | ■ is_active     | ■ run_num    |
| ■ delete_userid | ■ is_currver    | ■ state      |
| ■ description   | ■ joid          | ■ tag        |
| ■ domain        | ■ mach_name     | ■ wf_name    |
|                 | ■ owner         | ■ wf_ver     |

## ujointerface

The ujo\_zos\_condcodes table stores the information specified in the condition\_code attribute of z/OS Regular (ZOS) job definitions.

This table contains the following columns:

- |             |             |
|-------------|-------------|
| ■ action    | ■ program   |
| ■ job_name  | ■ rc        |
| ■ job_ver   | ■ result    |
| ■ joid      | ■ seq_num   |
| ■ over_num  | ■ step_name |
| ■ proc_step |             |

## ujos\_zos\_dsn\_trigger

The ujos\_zos\_dsn\_trigger table stores z/OS Data Set Trigger (ZOSDST) job definitions.

This table contains the following columns:

- |                 |            |                |
|-----------------|------------|----------------|
| ■ dsn           | ■ job_ver  | ■ trigger_by   |
| ■ explicit      | ■ joid     | ■ trigger_on   |
| ■ ftp_direction | ■ over_num | ■ trigger_type |
| ■ ftp_host      | ■ renamed  | ■ updated      |
| ■ ftp_userid    | ■ strings  | ■ userid       |

## ujos\_zos\_job

The ujos\_zos\_job table stores the definitions for Z/OS Regular (ZOS) and z/OS Manual (ZOSM) jobs.

This table contains the following columns:

- |                   |               |              |
|-------------------|---------------|--------------|
| ■ authstring      | ■ jcl_library | ■ over_num   |
| ■ copy_generation | ■ jcl_member  | ■ searchbkwd |
| ■ copy_library    | ■ job_name    | ■ zcondcodes |
| ■ copy_type       | ■ job_ver     |              |
| ■ envvars         | ■ joid        |              |



# Index

---

## A

ACTIVATED (9) status • 906  
after\_time attribute • 871  
agent local security • 542  
agent\_name attribute • 821  
ALARM (106) event • 896  
alarm attribute • 873  
alarm\_if\_fail attribute • 295  
alarm\_verif attribute • 874  
alarms  
    AUTO\_PING (526) • 910  
    CHASE (514) • 910  
    CPI\_JOBNAME\_INVALID (535) • 911  
    CPI\_UNAVAILABLE (536) • 911  
    DATABASE\_COMM (516) • 911  
    DB\_PROBLEM (523) • 911  
    DB\_ROLLOVER (519) • 911  
    DUPLICATE\_EVENT (524) • 911  
    EP\_HIGH\_AVAIL (522) • 912  
    EP\_SHUTDOWN (521) • 912  
    EVENT\_HDLR\_ERROR (507) • 912  
    EVENT\_QUE\_ERROR (508) • 912  
    EXTERN\_DEPS\_ERROR (529) • 912  
    FORKFAIL (501) • 913  
    INSTANCE\_UNAVAILABLE (525) • 913  
    JOBFAILURE (503) • 913  
    JOBNOT\_ONICEHOLD (509) • 913  
    KILLJOBFAIL (540) • 913  
    MACHINE\_UNAVAILABLE (532) • 913  
    MAX\_RETRY (505) • 913  
    MAXRUNALARM (510) • 914  
    MINRUNALARM (502) • 914  
    MISSING\_HEARTBEAT (513) • 914  
    MUST\_COMPLETE\_ALARM (538) • 914  
    MUST\_START\_ALARM (537) • 914  
    REPLY\_RESPONSE\_FAIL (542) • 914  
    RESOURCE (512) • 915  
    RESTARTJOBFAIL (544) • 915  
    RETURN\_RESOURCE\_FAIL (543) • 915  
    SENDSIGFAIL (541) • 915  
    SERVICEDESK\_FAILURE (533) • 915  
    STARTJOBFAIL (506) • 915  
    UNINOTIFY\_FAILURE (534) • 916  
    VERSION\_MISMATCH (518) • 916

WAIT\_REPLY\_ALARM (539) • 916  
all\_events attribute • 875  
all\_status attribute • 876  
amount attribute • 844  
application attribute • 296  
arc\_obj\_name attribute • 297  
arc\_obj\_variant attribute • 298  
arc\_parms attribute • 299  
archive\_events command • 30  
archiving files • 33  
as\_config command • 37  
as\_info command • 38  
as\_safetool command • 40  
as\_server command • 47  
as\_test command • 48  
astail command • 49  
audience • 23  
auth\_string attribute • 303  
auto\_delete attribute • 304  
auto\_hold attribute • 305  
AUTO\_PING (526) alarm • 910  
auto\_svcdesk command • 50  
autoaggr command • 54  
autocal\_asc command • 55  
autoflags command • 69  
autoping command • 71  
autorep command  
    columns • 82  
    overview • 75  
    types of reports • 82  
    usage notes • 81  
autosec\_test command • 96  
autostatad command • 105  
autostatus command • 100  
autosys\_secure  
    database password • 111  
    EDIT superuser and EXEC superuser • 110  
    encrypted passwords • 112  
    native and embedded security • 110  
    overview • 107  
    remote authentication method • 111  
    running autosys\_secure • 112  
    user and host management • 111  
autosyslog command • 114  
autotimezone command • 118

---

TZ variable syntax • 121  
autotrack command • 126  
avg\_runtime attribute • 306

## B

bdc\_err\_rate attribute • 307  
bdc\_ext\_log attribute • 308  
bdc\_proc\_rate attribute • 309  
bdc\_system attribute • 310  
bean\_name attribute • 311  
blob • 887  
blob\_file attribute • 312, 891  
blob\_input attribute • 313, 891  
blob\_mode attribute • 881, 892  
blob\_type attribute • 893  
box\_failure attribute • 314  
box\_name attribute • 316  
box\_success attribute • 317  
box\_terminator attribute • 319

## C

calendars  
extended calendars • 61  
standard calendars • 61  
CHANGE\_PRIORITY (120) event • 897  
CHANGE\_STATUS (101) event • 897  
changing owner • 542  
character\_code attribute • 823  
CHASE (514) alarm • 910  
chase command • 135, 137  
CHECK\_HEARTBEAT (116) event • 898  
chk\_auto\_up command • 138  
CHK\_BOX\_TERM (118) event • 898  
chk\_files attribute • 320  
CHK\_MAX\_ALARM (114) event • 898  
CHK\_RUN\_WINDOW (122) event • 898  
class\_name attribute • 322  
clean\_files command • 155  
command attribute  
examples • 328, 333  
overview • 323, 807  
UNIX considerations • 325  
Windows considerations • 330  
commands  
archive\_events • 30  
archive\_jobs • 36  
as\_config • 37  
as\_info • 38

as\_safetool • 40  
as\_server • 47  
as\_test • 48  
astail • 49  
auto\_svcdesk • 50  
autoaggr • 54  
autocal\_asc • 55  
autoflags • 69  
autoping • 71  
autoprofm • 73  
autorep • 75  
autosec\_test • 96  
autostataid • 105  
autostatus • 100  
autosys\_secure • 107  
autosyslog • 114  
autotimezone • 118  
autotrack • 126  
chase • 135  
chk\_auto\_up • 138  
clean\_files • 155  
cron2jil • 157  
DBMaint • 160  
dbspace • 159  
dbstatistics • 162  
event\_demon • 163  
eventor • 164  
forecast command • 166  
initautosys command • 169  
jil • 170  
job\_depends • 176  
monbro • 180  
sendevent • 183  
time0 • 198  
wildcard characters • 28  
COMMENT (117) event • 899  
common job attributes • 293  
condition attribute • 335  
cross-instance job dependencies • 340  
exit code dependencies • 342  
global variable dependencies • 343  
job status dependencies • 337  
look-back dependencies • 344  
condition\_code attribute • 346  
connect\_string attribute • 349  
connection\_factory attribute • 351  
continuous attribute • 351  
copy\_jcl attribute • 356  
CPI\_JOBNAME\_INVALID (535) alarm • 911

---

CPI\_UNAVAILABLE (536) alarm • 911  
cpu\_usage attribute • 360  
CREATE file trigger type • 757  
create\_method attribute • 357  
create\_name attribute • 358  
create\_parameter attribute • 359  
cron2jil command • 157  
currun attribute • 877

## D

database tables and views  
  uko\_afm • 919  
  uko\_afm\_strings • 920  
  uko\_agent\_alias • 920  
  uko\_alamode table • 920  
  uko\_alarm table • 921  
  uko\_asbnode table • 921  
  uko\_asext\_config table • 922  
  uko\_audit\_info table • 922  
  uko\_audit\_msg table • 923  
  uko\_avg\_job\_runs table • 923  
  uko\_calendar table • 923  
  uko\_chase table • 924  
  uko\_chkpkt\_rstart table • 924  
  uko\_comm\_recv\_seq • 925  
  uko\_comm\_send\_seq • 925  
  uko\_command\_job • 926  
  uko\_cred table • 926  
  uko\_cycle table • 927  
  uko\_event table • 927  
  uko\_event2 table • 929  
  uko\_eventvu view • 929  
  uko\_ext\_calendar table • 930  
  uko\_ext\_event • 930  
  uko\_ext\_job table • 932  
  uko\_extended\_jobrun\_info • 932  
  uko\_file\_watch\_job • 933  
  uko\_ftp\_job • 933  
  uko\_glob table • 932  
  uko\_globblob table • 934  
  uko\_ha\_process table • 934  
  uko\_i5\_job • 935  
  uko\_intcodes table • 935  
  uko\_job table • 936  
  uko\_job\_cond table • 937  
  uko\_job\_resource\_dep • 938  
  uko\_job\_runs table • 938  
  uko\_job\_status table • 939

uko\_job\_tree • 939  
uko\_jobblob table • 940  
uko\_jobst view • 941  
uko\_jobtype table • 942  
uko\_keymaster table • 942  
uko\_last\_Eoid\_counter table • 942  
uko\_ma\_process • 943  
uko\_machine table • 943  
uko\_meta\_enumerations • 944  
uko\_meta\_properties • 944  
uko\_meta\_rules • 945  
uko\_meta\_types • 945  
uko\_micro\_focus\_job • 946  
uko\_monbro table • 946  
uko\_monitor\_object\_job • 947  
uko\_monitor\_winevent\_job • 947  
uko\_msg\_ack table • 948  
uko\_next\_oid table • 948  
uko oraaps • 949  
uko oraaps\_steps • 949  
uko\_patch • 950  
uko\_peoplesoft\_job • 950  
uko\_proc\_event table • 951  
uko\_real\_resource • 952  
uko\_rep\_daily table • 953  
uko\_rep\_hourly table • 954  
uko\_rep\_monthly table • 955  
uko\_rep\_weekly table • 956  
uko\_req\_job table • 957  
uko\_reswait\_que • 957  
uko\_sap\_arcspec • 958  
uko\_sap\_infodetail • 958  
uko\_sap\_job • 959  
uko\_sap\_jobstep • 959  
uko\_sap\_prtspec • 960  
uko\_sap\_recipient • 961  
uko\_sched\_info • 962  
uko\_service\_desk • 962  
uko\_snmp\_job • 963  
uko\_sql\_job • 963  
uko\_sql\_proc\_parms • 964  
uko\_strings • 964  
uko\_sys\_ha\_state table • 964  
uko\_temp\_jobruns • 964  
uko\_timezones table • 965  
uko\_uninotify table • 965  
uko\_virt\_resource • 964  
uko\_virt\_resource\_lookup • 966  
uko\_virt\_resource\_status • 966

---

uko\_wait\_que table • 967  
uko\_web\_services • 968  
uko\_web\_services2 • 969  
uko\_wol\_job • 969  
uko\_workflow • 970  
uko\_zos\_condcodes • 970  
uko\_zos\_dsn\_trigger • 971  
uko\_zos\_job • 971  
DATABASE\_COMM (516) alarm • 911  
date\_conditions attribute • 362  
days\_of\_week attribute • 363  
DB\_PROBLEM (523) alarm • 911  
DB\_ROLLOVER (519) alarm • 911  
DBMaint command • 160  
dbspace command • 159  
dbstatistics command • 162  
dbtype attribute • 365  
DELETE file trigger type • 758  
delete\_blob subcommand • 887  
delete\_box subcommand • 287  
delete\_glob subcommand • 888  
delete\_job subcommand • 288  
delete\_job\_type subcommand • 805  
delete\_machine subcommand • 809  
delete\_monbro subcommand • 867  
delete\_resource subcommand • 839  
delete\_xinst subcommand • 855  
DELETEJOB (119) event • 899  
deleting  
    all references to a real machine and the machine  
        itself • 812  
    real machine • 810  
    real machine pool • 811  
    reference to a real machine • 811  
    virtual machine • 811  
dependencies (job)  
    cross-instance • 340  
    exit code • 342  
    global variables • 343  
    job status • 337  
    look-back • 344  
description attribute • 366, 808, 824, 845  
destination\_file attribute • 367  
destination\_name attribute • 369  
disk\_drive attribute • 371  
disk\_format attribute • 372  
disk\_space attribute • 374  
DUPLICATE\_EVENT (524) alarm • 911

## E

encoding attribute • 375  
encryption\_type attribute • 824  
endpoint\_URL attribute • 376  
envvars attribute • 378  
EP\_HIGH\_AVAIL (522) alarm • 912  
EP\_ROLLOVER (520) alarm • 912  
EP\_SHUTDOWN (521) alarm • 912  
event\_demon command • 163  
EVENT\_HDLR\_ERROR (507) alarm • 912  
EVENT\_QUE\_ERROR (508) alarm • 912  
eventor command  
    log files • 165  
    overview • 164  
events  
    ALARM (106) • 896  
    ALERT (139) • 897  
    CHANGE\_PRIORITY (120) • 897  
    CHANGE\_STATUS (101) • 897  
    CHECK\_HEARTBEAT (116) • 898  
    CHK\_BOX\_TERM (118) • 898  
    CHK\_COMPLETE (136) • 898  
    CHK\_RUN\_WINDOW (122) • 898  
    CHK\_START (135) • 898  
    CHK\_TERM\_RUNTIME (138) • 899  
    COMMENT (117) • 899  
    DELETEJOB (119) • 899  
    EXTERNAL\_DEPENDENCY (127) • 899  
    FORCE\_STARTJOB (108) • 900  
    HEARTBEAT (115) • 900  
    INVALIDATE\_MACH (133) • 900  
    JOB\_OFF\_HOLD (113) • 900  
    JOB\_OFF\_ICE (111) • 901  
    JOB\_ON\_HOLD (112) • 901  
    JOB\_ON\_ICE (110) • 901  
    KILLJOB (105) • 902  
    MACH\_OFFLINE (130) • 903  
    MACH\_ONLINE (131) • 903  
    MACH\_PROVISION (142) • 903  
    REFRESH\_EXTINST (129) • 904  
    RELEASE\_RESOURCE (137) • 904  
    REPLY\_RESPONSE (140) • 904  
    RESTARTJOB (143) • 904  
    SEND\_SIGNAL (126) • 905  
    SET\_GLOBAL (125) • 904  
    STARTJOB (107) • 905  
    STATE\_CHANGE (141) • 905  
    STOP\_DEMON (109) • 905

---

## examples

- as\_safetool command • 42
- autorep command • 85
- exclude\_calendar attribute • 380
- EXIST file trigger type • 758
- exit codes
  - as\_safetool command • 46
  - system states • 917
- EXPAND file trigger type • 759
- EXTERN\_DEPS\_ERROR (529) alarm • 912
- EXTERNAL\_DEPENDENCY (127) event • 899

## F

- factor attribute • 826
- fail\_codes attribute • 381
- FAILURE (5) status • 907
- failure attribute • 878
- file trigger notes • 741
- file trigger type notes
  - CREATE • 757
  - DELETE • 758
  - EXIST • 758
  - EXPAND • 759
  - GENERATE • 768
  - NOTEXIST • 762
  - SHRINK • 763
  - UPDATE • 767
- filter attribute • 383
- filter\_type attribute • 385
- finder\_name attribute • 386
- finder\_parameter attribute • 387
- force attribute • 828
- FORCE\_STARTJOB (108) event • 900
- FORKFAIL (501) alarm • 913
- ftp\_command attribute • 389
- ftp\_compression attribute • 390
- ftp\_local\_name attribute • 392
- ftp\_local\_user attribute • 395
- ftp\_remote\_name attribute • 396
- ftp\_server\_name attribute • 400
- ftp\_server\_port attribute • 402
- ftp\_transfer\_direction attribute • 403
- ftp\_transfer\_type attribute • 404
- ftp\_use\_SSL attribute • 406

## G

- glob • 887
- group attribute • 409

## H

- HEARTBEAT (115) event • 900
- heartbeat\_interval attribute • 410

## I

- i5\_action attribute • 411
- i5\_cc\_exit attribute • 413
- i5\_curr\_lib attribute • 415
- i5\_lda attribute • 419
- i5\_job\_desc attribute • 416
- i5\_job\_name attribute • 417
- i5\_job\_queue attribute • 418
- i5\_lib attribute • 420
- i5\_library\_list attribute • 421
- i5\_name attribute • 423
- i5\_others attribute • 426
- i5\_params attribute • 427
- i5\_process\_priority attribute • 428
- INACTIVE (8) status • 907
- initial\_context\_factory attribute • 429
- insert\_blob subcommand • 888
- insert\_glob subcommand • 890
- insert\_job subcommand • 289
- insert\_job\_type subcommand • 806
- insert\_machine subcommand • 813
- insert\_monbro subcommand • 868
- insert\_resource subcommand • 840
- insert\_xinst subcommand • 856
- inserting
  - real machine pools • 819
  - real machines • 817
  - virtual machines • 818
- inside\_range attribute • 431
- INSTANCE\_UNAVAILABLE (525) alarm • 913
- interactive attribute • 433
- introduction • 23
- invocation\_type attribute • 434
- ip\_host attribute • 435
- ip\_port attribute • 437
- ip\_status attribute • 438
- issuing commands
  - on UNIX • 27
  - on Windows • 28

## J

- j2ee\_authentication\_order attribute • 439
- j2ee\_conn\_domain attribute • 440

---

j2ee\_conn\_origin attribute • 442  
j2ee\_no\_global\_proxy\_defaults attribute • 443  
j2ee\_parameter attribute • 444  
j2ee\_proxy\_domain attribute • 446  
j2ee\_proxy\_host attribute • 447  
j2ee\_proxy\_origin\_host attribute • 449  
j2ee\_proxy\_port attribute • 450  
j2ee\_proxy\_user attribute • 451  
j2ee\_user attribute • 452  
jcl\_library attribute • 454  
jcl\_member attribute • 455  
JIL  
    syntax rules • 25  
JIL blob and glob definitions  
    blob\_file attribute • 891  
    blob\_input attribute • 891  
    blob\_mode attribute • 892  
    blob\_type attribute • 893  
    delete\_blob subcommand • 887  
    delete\_glob subcommand • 888  
    insert\_blob subcommand • 888  
    insert\_glob subcommand • 890  
JIL command • 170  
    JIL syntax rules • 173  
JIL cross-instance definitions  
    delete\_xinst subcommand • 855  
    insert\_xinst subcommand • 856  
    update\_xinst subcommand • 858  
    xcrypt\_type attribute • 860  
    xkey\_to\_manager attribute • 861  
    xmachine attribute • 862  
    xmanager attribute • 863  
    xport attribute • 864  
    xtype attribute • 865  
JIL job type definitions  
    command attribute • 807  
    delete\_job\_type subcommand • 805  
    description attribute • 808  
    insert\_job\_type subcommand • 806  
    update\_job\_type subcommand • 807  
JIL machine definitions  
    agent\_name attribute • 821  
    character\_code attribute • 823  
    delete\_machine subcommand • 809  
    description attribute • 824  
    encryption\_type attribute • 824  
    factor attribute • 826  
    force attribute • 828  
    insert\_machine subcommand • 813  
key\_to\_agent attribute • 829  
machine attribute • 830  
max\_load attribute • 831  
node\_name attribute • 833  
opsys attribute • 833  
port attribute • 835  
remove\_references attribute • 836  
type attribute • 837  
update\_machine subcommand • 820  
JIL monitor or report definitions  
    after\_time attribute • 871  
    alarm attribute • 873  
    alarm\_verif attribute • 874  
    all\_events attribute • 875  
    all\_status attribute • 876  
    currun attribute • 877  
    delete\_monbro subcommand • 867  
    failure attribute • 878  
    insert\_monbro subcommand • 868  
    job\_filter attribute • 879  
    job\_name attribute • 880  
    mode attribute • 881  
    restart attribute • 882  
    running attribute • 883  
    starting attribute • 884  
    success attribute • 885  
    terminated attribute • 886  
    update\_monbro subcommand • 870  
JIL resource definitions  
    amount attribute • 844  
    delete\_resource subcommand • 839  
    description attribute • 845  
    insert\_resource subcommand • 840  
    machine attribute • 846  
    res\_type attribute • 847  
    resources attribute • 849  
    update\_resource subcommand • 842  
    jmx\_parameter attribute • 456  
    jmx\_user attribute • 457  
job definitions  
    alarm\_if\_fail attribute • 295  
    application attribute • 296  
    arc\_obj\_name attribute • 297  
    arc\_obj\_variant attribute • 298  
    arc\_parms attribute • 299  
    auth\_string attribute • 303  
    auto\_delete attribute • 304  
    auto\_hold attribute • 305  
    avg\_runtime attribute • 306

- 
- bdc\_err\_rate attribute • 307  
bdc\_ext\_log attribute • 308  
bdc\_proc\_rate attribute • 309  
bdc\_system attribute • 310  
bean\_name attribute • 311  
blob\_file attribute • 312  
blob\_input attribute • 313  
box\_failure attribute • 314  
box\_name attribute • 316  
box\_success attribute • 317  
box\_terminator attribute • 319  
chk\_files attribute • 320  
class\_name attribute • 322  
command attribute • 323  
condition attribute • 335  
condition\_code attribute • 346  
connect\_string attribute • 349  
connection\_factory attribute • 351  
continuous attribute • 351  
copy\_jcl attribute • 356  
cpu\_usage attribute • 360  
create\_method attribute • 357  
create\_name attribute • 358  
create\_parameter attribute • 359  
date\_conditions attribute • 362  
days\_of\_week attribute • 363  
dbtype attribute • 365  
delete\_box subcommand • 287  
delete\_job subcommand • 288  
description attribute • 366  
destination\_file attribute • 367  
destination\_name attribute • 369  
disk\_drive attribute • 371  
disk\_format attribute • 372  
disk\_space attribute • 374  
encoding attribute • 375  
endpoint\_URL attribute • 376  
envvars attribute • 378  
exclude\_calendar attribute • 380  
fail\_codes attribute • 381  
filter attribute • 383  
filter\_type attribute • 385  
finder\_name attribute • 386  
finder\_parameter attribute • 387  
ftp\_command attribute • 389  
ftp\_compression attribute • 390  
ftp\_local\_name attribute • 392  
ftp\_local\_user attribute • 395  
ftp\_remote\_name attribute • 396  
ftp\_server\_name attribute • 400  
ftp\_server\_port attribute • 402  
ftp\_transfer\_direction attribute • 403  
ftp\_transfer\_type attribute • 404  
ftp\_use\_SSL attribute • 406  
group attribute • 409  
heartbeat\_interval attribute • 410  
i5\_action attribute • 411  
i5\_cc\_exit attribute • 413  
i5\_curr\_lib attribute • 415  
i5\_lda attribute • 419  
i5\_job\_desc attribute • 416  
i5\_job\_name attribute • 417  
i5\_job\_queue attribute • 418  
i5\_lib attribute • 420  
i5\_library\_list attribute • 421  
i5\_name attribute • 423  
i5\_others attribute • 426  
i5\_params attribute • 427  
i5\_process\_priority attribute • 428  
initial\_context\_factory attribute • 429  
insert\_job subcommand • 289  
inside\_range attribute • 431  
interactive attribute • 433  
invocation\_type attribute • 434  
ip\_host attribute • 435  
ip\_port attribute • 437  
ip\_status attribute • 438  
j2ee\_authentication\_order attribute • 439  
j2ee\_conn\_domain attribute • 440  
j2ee\_conn\_origin attribute • 442  
j2ee\_no\_global\_proxy\_defaults attribute • 443  
j2ee\_parameter attribute • 444  
j2ee\_proxy\_domain attribute • 446  
j2ee\_proxy\_host attribute • 447  
j2ee\_proxy\_origin\_host attribute • 449  
j2ee\_proxy\_port attribute • 450  
j2ee\_proxy\_user attribute • 451  
j2ee\_user attribute • 452  
jcl\_library attribute • 454  
jcl\_member attribute • 455  
jmx\_parameter attribute • 456  
jmx\_user attribute • 457  
job\_class attribute • 458  
job\_load attribute • 460  
job\_terminator attribute • 461  
job\_type attribute • 464  
lower\_boundary attribute • 467, 470  
machine attribute • 473

---

max\_exit\_success attribute • 475  
max\_run\_alarm attribute • 476  
mbean\_attr attribute • 477  
mbean\_name attribute • 478  
mbean\_operation attribute • 480  
message\_class attribute • 481  
method\_name attribute • 482  
min\_run\_alarm attribute • 484  
modify\_parameter attribute • 485  
monitor\_cond attribute • 487  
monitor\_mode attribute • 488  
monitor\_type attribute • 492  
must\_complete\_times attribute • 493  
must\_start\_times attribute • 499  
n\_retrys attribute • 505  
no\_change attribute • 507  
notification\_id attribute • 509  
notification\_msg attribute • 510  
one\_way attribute • 510  
operation\_type attribute • 512  
oracle\_appl\_name attribute • 513  
oracle\_appl\_name\_type attribute • 515  
oracle\_args attribute • 516  
oracle\_desc attribute • 518  
oracle\_mon\_children attribute • 519  
oracle\_mon\_children\_delay attribute • 520  
oracle\_print\_copies attribute • 522  
oracle\_print\_style attribute • 523  
oracle\_printer attribute • 525  
oracle\_program attribute • 527  
oracle\_programdata attribute • 528  
oracle\_req\_set attribute • 530  
oracle\_resp attribute • 531  
oracle\_save\_output attribute • 533  
oracle\_use\_arg\_def attribute • 534  
oracle\_user attribute • 536  
override\_job subcommand • 290  
owner attribute • 538  
permission attribute • 543  
poll\_interval attribute • 547  
port\_name attribute • 549  
priority attribute • 551  
process\_name attribute • 552  
process\_status attribute • 555  
profile attribute • 557  
provider\_url attribute • 560, 563  
ps\_args attribute • 565  
ps\_dest\_format attribute • 566  
ps\_dest\_type attribute • 568  
ps\_detail\_folder attribute • 570  
ps\_dlist\_roles attribute • 571  
ps\_dlist\_users attribute • 572  
ps\_email\_address attribute • 575  
ps\_email\_log attribute • 577  
ps\_email\_subject attribute • 578  
ps\_email\_text attribute • 579  
ps\_email\_web\_report attribute • 580  
ps\_output\_dest attribute • 582  
ps\_process\_name attribute • 585  
ps\_process\_type attribute • 586  
ps\_restarts attribute • 587  
ps\_run\_cntrl\_args attribute • 588  
ps\_run\_cntrl\_id attribute • 591  
ps\_run\_control\_table attribute • 592  
ps\_server\_name attribute • 594  
ps\_skip\_parm\_updates attributes • 595  
ps\_time\_zone attribute • 597  
remote\_name attribute • 598  
request\_id attribute • 599  
result\_type attribute • 600  
return\_class\_name attribute • 601  
return\_namespace attribute • 602  
return\_xml\_name attribute • 604  
run\_calendar attribute • 606  
run\_window attribute • 607  
sap\_abap\_name attribute • 609  
sap\_chain\_id attribute • 610  
sap\_client attribute • 611  
sap\_event\_id attribute • 612  
sap\_event\_parm attribute • 613  
sap\_ext\_table attribute • 614  
sap\_fail\_msg attribute • 615  
sap\_info\_pack attribute • 616  
sap\_is\_trigger attribute • 617  
sap\_job\_class attribute • 618  
sap\_job\_count attribute • 619  
sap\_job\_name attribute • 620  
sap\_lang\_attribute • 622  
sap\_mon\_child attribute • 623  
sap\_office attribute • 624  
sap\_print\_parms attribute • 625  
sap\_proc\_type attribute • 631  
sap\_proc\_user attribute • 632  
sap\_process\_client attribute • 629  
sap\_process\_status attribute • 630  
sap\_recipients attribute • 633  
sap\_release\_option attribute • 635  
sap\_rfc\_dest attribute • 636

---

sap\_step\_num attribute • 637  
sap\_step\_parms attribute • 638  
sap\_success\_msg attribute • 647  
sap\_target\_jobname attribute • 648  
sap\_target\_sys attribute • 649  
scp\_local\_name attribute • 650  
scp\_local\_user attribute • 652  
scp\_protocol attribute • 653  
scp\_remote\_dir attribute • 654  
scp\_remote\_name attribute • 656  
scp\_server\_name attribute • 658  
scp\_server\_port attribute • 659  
scp\_target\_os attribute • 660  
scp\_transfer\_direction attribute • 661  
search\_bw attribute • 662  
send\_notification attribute • 664  
service\_desk attribute • 665  
service\_name attribute • 666  
shell attribute • 668  
sp\_arg attribute • 670  
sp\_name attribute • 673  
sql\_command attribute • 674  
start\_mins attribute • 676  
start\_times attribute • 677  
std\_err\_file attribute • 679  
std\_in\_file attribute • 683  
std\_out\_file attribute • 687  
success\_codes attribute • 692  
success\_criteria attribute • 693  
success\_pattern attribute • 696  
svcdesk\_attr attribute • 698  
svcdesk\_desc attribute • 700  
svcdesk\_imp attribute • 701  
svcdesk\_pri attribute • 702  
svcdesk\_sev attribute • 703  
tablename attribute • 704  
target\_namespace attribute • 705  
term\_run\_time attribute • 706  
text\_file\_filter attribute • 707  
text\_file\_filter\_exists attribute • 710  
text\_file\_mode attribute • 711  
text\_file\_name attribute • 715  
time\_format attribute • 717  
time\_position attribute • 720  
timezone attribute • 722  
trigger\_cond attribute • 724  
trigger\_type attribute • 726  
ulimit attribute • 729  
update\_job subcommand • 292  
upper\_boundary attribute • 731, 734  
URL attribute • 736  
use\_topic attribute • 737  
user\_role attribute • 738  
watch\_file attribute • 740  
watch\_file\_change\_type attribute • 746  
watch\_file\_change\_value attribute • 748  
watch\_file\_groupname attribute • 750  
watch\_file\_min\_size attribute • 751  
watch\_file\_owner attribute • 752  
watch\_file\_recursive attribute • 754  
watch\_file\_type attribute • 756  
watch\_file\_win\_user attribute • 769  
watch\_interval attribute • 770  
watch\_no\_change attribute • 772  
web\_parameter attribute • 773  
web\_user attribute • 776  
win\_event\_category attribute • 777  
win\_event\_computer attribute • 778  
win\_event\_datetime attribute • 779  
win\_event\_description attribute • 780  
win\_event\_id attribute • 781  
win\_event\_op attribute • 782  
win\_event\_source attribute • 783  
win\_event\_type attribute • 784  
win\_log\_name attribute • 785  
win\_service\_name attribute • 786  
win\_service\_status attribute • 787  
wsdl\_operation attribute • 788  
WSDL\_URL attribute • 790  
zos\_dataset attribute • 792  
zos\_dsn\_renamed attribute • 793  
zos\_dsn\_updated attribute • 794  
zos\_explicit\_dsn attribute • 795  
zos\_ftp\_direction attribute • 796  
zos\_ftp\_host attribute • 797  
zos\_ftp\_userid attribute • 798  
zos\_jobname attribute • 799  
zos\_trigger\_by attribute • 800  
zos\_trigger\_on attribute • 802  
zos\_trigger\_type attribute • 803  
job owners • 542  
job types  
    box jobs • 201  
    command jobs • 204  
    CPU monitoring jobs • 206  
    database monitor jobs • 207  
    database stored procedure jobs • 209  
    database trigger jobs • 211

---

disk monitoring jobs • 213  
entity bean jobs • 214  
file trigger jobs • 217  
file watcher jobs • 219  
FTP jobs • 220  
HTTP jobs • 222  
i5/OS jobs • 224  
IP monitoring jobs • 226  
JMS publish jobs • 228  
JMS subscribe jobs • 230  
JMX-MBean attribute get jobs • 232  
JMX-MBean attribute set jobs • 233  
JMX-MBean create instance jobs • 235  
JMX-MBean operation jobs • 237  
JMX-MBean remove instance jobs • 239  
JMX-MBean subscribe jobs • 240  
Oracle E-Business Suite copy single request jobs • 242  
Oracle E-Business Suite request set jobs • 244  
Oracle E-Business Suite single request jobs • 246  
overview • 201  
PeopleSoft jobs • 249  
POJO jobs • 252  
process monitoring jobs • 253  
RMI jobs • 254  
SAP batch input session jobs • 255  
SAP BW infopackage jobs • 256  
SAP BW process chain jobs • 258  
SAP data archiving jobs • 259  
SAP event monitor jobs • 261  
SAP job copy jobs • 262  
SAP process monitor jobs • 264  
SAP R/3 jobs • 266  
secure copy jobs • 268  
session bean jobs • 270  
SQL jobs • 272  
text file reading and monitoring jobs • 274  
web services jobs • 276  
Windows event log monitoring jobs • 279  
Windows service monitoring jobs • 281  
z/OS data set trigger jobs • 282  
z/OS manual jobs • 284  
z/OS regular jobs • 285  
job\_class attribute • 458  
job\_depends command • 176  
job\_filter attribute • 879  
job\_load attribute • 460  
job\_name attribute • 880  
JOB\_OFF\_HOLD (113) event • 900

JOB\_OFF\_ICE (111) event • 901  
JOB\_ON\_HOLD (112) event • 901  
JOB\_ON\_ICE (110) event • 901  
job\_terminator attribute • 461  
job\_type attribute • 464  
JOBFAILURE (503) alarm • 913  
JOBNOT\_ONICEHOLD (509) alarm • 913

## K

key\_to\_agent attribute • 829  
KILLJOB (105) event • 902

## L

legacy agents • 23  
lower\_boundary attribute • 467, 470

## M

MACH\_OFFLINE (130) event • 903  
MACH\_ONLINE (131) event • 903  
machine attribute • 473, 830, 846  
MACHINE\_UNAVAILABLE (532) alarm • 913  
max\_exit\_success attribute • 475  
max\_load attribute • 831  
MAX\_RETRY (505) alarm • 913  
max\_run\_alarm attribute • 476  
MAXRUNALARM (510) alarm • 914  
mbean\_attr attribute • 477  
mbean\_name attribute • 478  
mbean\_operation attribute • 480  
message\_class attribute • 481  
method\_name attribute • 482  
min\_run\_alarm attribute • 484  
MINRUNALARM (502) alarm • 914  
MISSING\_HEARTBEAT (513) alarm • 914  
modify\_parameter attribute • 485  
monbro command • 180  
monitor\_cond attribute • 487  
monitor\_mode attribute • 488  
monitor\_type attribute • 492  
must\_complete\_times attribute • 493  
must\_start\_times attribute • 499

## N

n\_retrys attribute • 505  
no\_change attribute • 507  
node\_name attribute • 833  
NOTEXIST file trigger type • 762  
notification\_id attribute • 509

---

notification\_msg attribute • 510

## O

ON\_HOLD (11) status • 907  
ON\_ICE (7) status • 907  
one\_way attribute • 510  
operation\_type attribute • 512  
opsys attribute • 833  
oracle\_appl\_name attribute • 513  
oracle\_appl\_name\_type attribute • 515  
oracle\_args attribute • 516  
oracle\_desc attribute • 518  
oracle\_mon\_children attribute • 519  
oracle\_mon\_children\_delay attribute • 520  
oracle\_print\_copies attribute • 522  
oracle\_print\_style attribute • 523  
oracle\_printer attribute • 525  
oracle\_program attribute • 527  
oracle\_programdata attribute • 528  
oracle\_req\_set attribute • 530  
oracle\_resp attribute • 531  
oracle\_save\_output attribute • 533  
oracle\_use\_arg\_def attribute • 534  
oracle\_user attribute • 536  
override\_job subcommand • 290  
overview • 107  
owner attribute  
    changing the owner in multiple jobs • 542  
    job owners and agent local security • 542  
    overview • 538

## P

PEND\_MACH (14) • 907  
permission attribute  
    job permissions and Windows • 547  
    overview • 543  
    permission types on UNIX • 545  
    permission types on Windows • 546  
    user and permission types on UNIX • 546  
    user types • 544  
permissions  
    types • 545, 546  
    user types • 544  
    Windows NT • 547  
poll\_interval attribute • 547  
port attribute • 835  
port\_name attribute • 549  
priority attribute • 551

process\_name attribute • 552  
process\_status attribute • 555  
profile attribute • 557  
provider\_url attribute • 560, 563  
ps\_args attribute • 565  
ps\_dest\_format attribute • 566  
ps\_dest\_type attribute • 568  
ps\_detail\_folder attribute • 570  
ps\_dlist\_roles attribute • 571  
ps\_dlist\_users attribute • 572  
ps\_email\_address attribute • 575  
ps\_email\_log attribute • 577  
ps\_email\_subject attribute • 578  
ps\_email\_text attribute • 579  
ps\_email\_web\_report attribute • 580  
ps\_output\_dest attribute • 582  
ps\_process\_name attribute • 585  
ps\_process\_type attribute • 586  
ps\_restarts attribute • 587  
ps\_run\_ctrl\_args attribute • 588  
ps\_run\_ctrl\_id attribute • 591  
ps\_run\_control\_table attribute • 592  
ps\_server\_name attribute • 594  
ps\_skip\_parm\_updates attributes • 595  
ps\_time\_zone attribute • 597

## Q

QUE\_WAIT (12) status • 908

## R

real machines  
    deleting • 810  
    overview • 817  
REFRESH\_EXTINST (129) event • 904  
remote\_name attribute • 598  
remove\_references attribute • 836  
request\_id attribute • 599  
res\_type attribute • 847  
RESOURCE (512) alarm • 915  
resources attribute • 849  
RESTART (10) status • 908  
restart attribute • 882  
result\_type attribute • 600  
return\_class\_name attribute • 601  
return\_namespace attribute • 602  
return\_xml\_name attribute • 604  
run\_calendar attribute • 606  
run\_window attribute • 607

---

RUNNING (1) status • 908  
running attribute • 883  
running autosys\_secure  
    from the Command line • 112  
    windows agent user authentication • 113  
running chk\_auto\_up  
    with the -r arguments • 143  
    without any arguments • 140

## S

sap\_abap\_name attribute • 609  
sap\_chain\_id attribute • 610  
sap\_client attribute • 611  
sap\_event\_id attribute • 612  
sap\_event\_parm attribute • 613  
sap\_ext\_table attribute • 614  
sap\_fail\_msg attribute • 615  
sap\_info\_pack attribute • 616  
sap\_is\_trigger attribute • 617  
sap\_job\_class attribute • 618  
sap\_job\_count attribute • 619  
sap\_job\_name attribute • 620  
sap\_lang\_attribute • 622  
sap\_mon\_child attribute • 623  
sap\_office attribute • 624  
sap\_print\_parms attribute • 625  
sap\_proc\_type attribute • 631  
sap\_proc\_user attribute • 632  
sap\_process\_client attribute • 629  
sap\_process\_status attribute • 630  
sap\_recipients attribute • 633  
sap\_release\_option attribute • 635  
sap\_rfc\_dest attribute • 636  
sap\_step\_num attribute • 637  
sap\_step\_parms attribute • 638  
sap\_success\_msg attribute • 647  
sap\_target\_jobname attribute • 648  
sap\_target\_sys attribute • 649  
scp\_local\_name attribute • 650  
scp\_local\_user attribute • 652  
scp\_protocol attribute • 653  
scp\_remote\_dir attribute • 654  
scp\_remote\_name attribute • 656  
scp\_server\_name attribute • 658  
scp\_server\_port attribute • 659  
scp\_target\_os attribute • 660  
scp\_transfer\_direction attribute • 661  
search\_bw attribute • 662

send\_notification attribute • 664  
SEND\_SIGNAL (126) event • 905  
sendevent command • 183  
service\_desk attribute • 665  
service\_name attribute • 666  
SERVICEDESK\_FAILURE (533) alarm • 915  
SET\_GLOBAL (125) event • 904  
shell attribute • 668  
SHRINK file trigger type • 763  
sp\_arg attribute • 670  
sp\_name attribute • 673  
sql\_command attribute • 674  
start\_mins attribute • 676  
start\_times attribute • 677  
STARTING (3) status • 908  
starting attribute • 884  
STARTJOB (107) event • 905  
STARTJOBFAIL (506) alarm • 915  
status  
    ACTIVATED (9) • 906  
    FAILURE (5) • 907  
    INACTIVE (8) • 907  
    ON\_HOLD (11) • 907  
    ON\_ICE (7) • 907  
    PEND\_MACH (14) • 907  
    QUE\_WAIT (12) • 908  
    RESTART (10) • 908  
    RESWAIT (15) • 908  
    RUNNING (1) • 908  
    STARTING (3) • 908  
    SUCCESS (4) • 908  
    TERMINATED (6) • 909  
    WAIT\_REPLY (13) • 909  
std\_err\_file attribute • 679  
std\_in\_file attribute • 683  
std\_out\_file attribute • 687  
STOP\_DEMON (109) event • 905  
SUCCESS (4) status • 908  
success attribute • 885  
success\_codes attribute • 692  
success\_criteria attribute • 693  
success\_pattern attribute • 696  
supported data types • 672  
svcdesk\_attr attribute • 698  
svcdesk\_desc attribute • 700  
svcdesk\_imp attribute • 701  
svcdesk\_pri attribute • 702  
svcdesk\_sev attribute • 703  
syntax rules • 25

---

system states  
  alarms • 909  
  events • 895  
  exit codes • 917  
  status • 906

## T

tablename attribute • 704  
target\_namespace attribute • 705  
term\_run\_time attribute • 706  
TERMINATED (6) status • 909  
terminated attribute • 886  
text\_file\_filter attribute • 707  
text\_file\_filter\_exists attribute • 710  
text\_file\_mode attribute • 711  
text\_file\_name attribute • 715  
time\_format attribute • 717  
time\_position attribute • 720  
time0 command • 198  
timezone attribute • 722  
trigger\_cond attribute • 724  
trigger\_type attribute • 726  
type attribute • 837

## U

uko\_alamode table • 920  
uko\_alarm table • 921  
uko\_asbnode table • 921  
uko\_asext\_config table • 922  
uko\_audit\_info table • 922  
uko\_audit\_msg table • 923  
uko\_avg\_job\_runs table • 923  
uko\_calendar table • 923  
uko\_chase table • 924  
uko\_chkpkt\_rstart table • 924  
uko\_cred table • 926  
uko\_cycle table • 927  
uko\_event table • 927  
uko\_event2 table • 929  
uko\_eventvu view • 929  
uko\_ext\_calendar table • 930  
uko\_ext\_job table • 932  
uko\_glob table • 934  
uko\_globblob table • 934  
uko\_ha\_process table • 934  
uko\_intcodes table • 935  
uko\_job table • 936  
uko\_job\_cond table • 937

uko\_job\_runs table • 938  
uko\_job\_status table • 939  
uko\_jobblob table • 940  
uko\_jobst view • 941  
uko\_jobtype table • 942  
uko\_keymaster table • 942  
uko\_last\_Eoid\_counter table • 942  
uko\_machine table • 943  
uko\_monbro table • 946  
uko\_msgack table • 948  
uko\_next\_oid table • 948  
uko\_proc\_event table • 951  
uko\_rep\_daily table • 953  
uko\_rep\_hourly table • 954  
uko\_rep\_monthly table • 955  
uko\_rep\_weekly table • 956  
uko\_req\_job table • 957  
uko\_sys\_ha\_state table • 964  
uko\_timezones table • 965  
uko\_uninotify table • 965  
uko\_wait\_que table • 967  
ukoadmin • 159  
ulimit attribute • 729  
UNINOTIFY\_FAILURE (534) alarm • 916

## UNIX

  considerations for command attribute • 325  
  examples • 328  
unsupported data types • 673  
UPDATE file trigger type • 767  
update\_job subcommand • 292  
update\_job\_type subcommand • 807  
update\_machine subcommand • 820  
update\_monbro subcommand • 870  
update\_resource subcommand • 842  
update\_xinst subcommand • 858  
updating virtual machines • 821  
upper\_boundary attribute • 731, 734  
URL attribute • 736  
use\_topic attribute • 737  
user\_role attribute • 738

## V

VERSION\_MISMATCH (518) alarm • 916  
virtual machines  
  deleting • 811  
  overview • 818  
  updating • 821

---

## W

watch\_file attribute  
    overview • 740  
    specifying i5/OS files • 744  
    specifying UNIX files • 743  
    specifying Windows files • 743  
watch\_file attribute examples • 743, 744  
watch\_file\_change\_type attribute • 746  
watch\_file\_change\_value attribute • 748  
watch\_file\_groupname attribute • 750  
watch\_file\_min\_size attribute • 751  
watch\_file\_owner attribute • 752  
watch\_file\_recursive attribute • 754  
watch\_file\_type attribute • 756  
watch\_file\_win\_user attribute • 769  
watch\_interval attribute • 770  
watch\_no\_change attribute • 772  
web\_parameter attribute • 773  
web\_user attribute • 776  
wildcard characters, using in commands • 28  
win\_event\_category attribute • 777  
win\_event\_computer attribute • 778  
win\_event\_datetime attribute • 779  
win\_event\_description attribute • 780  
win\_event\_id attribute • 781  
win\_event\_op attribute • 782  
win\_event\_source attribute • 783  
win\_event\_type attribute • 784  
win\_log\_name attribute • 785  
win\_service\_name attribute • 786  
win\_service\_status attribute • 787  
Windows  
    considerations for the command attribute • 330  
    examples • 333  
wsdl\_operation attribute • 788  
WSDL\_URL attribute • 790

## X

xcrypt\_type attribute • 860  
xkey\_to\_manager attribute • 861  
xmachine attribute • 862  
xmanager attribute • 863  
xport attribute • 864  
xtype attribute • 865

## Z

zos\_dataset attribute • 792

zos\_dsn\_renamed attribute • 793  
zos\_dsn\_updated attribute • 794  
zos\_explicit\_dsn attribute • 795  
zos\_ftp\_direction attribute • 796  
zos\_ftp\_host attribute • 797  
zos\_ftp\_userid attribute • 798  
zos\_jobname attribute • 799  
zos\_trigger\_by attribute • 800  
zos\_trigger\_on attribute • 802  
zos\_trigger\_type attribute • 803