CA Integrated Agent Services

User Guide

r11

Second Edition

CA Technologies Product References

This document references the following CA Technologies products:

- CA Workload Automation Autosys Edition
- CA Workload Automation dSeries Edition
- CA Workload Automation ESP Edition
- CA Workload Automation CA 7® Edition
- CA Workload Automation Desktop Client (CA WA Desktop Client)
- CA Workload Automation Web Client (CA WA Web Client)
- CA Workload Automation High Availability DE (CA WA High Availability)
- CA Workload Automation Web Services (CA WA Web Services)
- 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 HP Integrity NonStop
- CA Workload Automation Agent for i5/OS (CA WA Agent for i5/OS)
- CA Workload Automation Agent for z/OS (CA WA Agent for z/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 Micro Focus (CA WA Agent for Micro Focus)
- 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 Restart Option ESP Edition (CA WA Restart Option ESP Edition)
- CA Spectrum® Service Assurance (CA Spectrum SA)

Contact CA Technologies

Contact CA Support

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

- 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

Providing Feedback About Product Documentation

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

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.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- CA Product References (see page 3)—Added CA Workload Automation Agent for HP Integrity NonStop
- Job Types—Added new job type NONSTOP_JOB for HP Integrity NonStop workload
- File Trigger Jobs (see page 87)—Added the HP Integrity NonStop system and CA WA Agent for HP Integrity NonStop to the description
- Defining File Trigger Jobs (see page 88)—Added example "Monitor an Update to a Payroll File Object on an HP Integrity NonStop System"
- HP Integrity NonStop Jobs (see page 105)—New, including all topics under this one.
- ARGS Statement—Pass Arguments (UNIX, Windows, and HP Integrity NonStop Jobs) (see page 257)
  - Renamed the statement from "ARGS Statement—Pass Arguments (UNIX and Windows Jobs)"
  - Added "HP Integrity NonStop" to the list of supported job types
  - Added example "Pass Positional Arguments to CA WA Agent for HP Integrity NonStop"
- ASSIGN Statement—Assign a Logical File Name to a Physical File (see page 262)—New
- COMMAND Statement—Run Agent Commands (see page 289)
  - Renamed the statement from "COMMAND Statement—Run i5/OS Command (i5/OS Jobs)".
  - Added "HP Integrity NonStop" to the list of supported job types
  - Added an HP Integrity NonStop description to the *command* operand.
  - Split the usage notes into i5/OS and HP Integrity NonStop sections.
  - Added a new alternate syntax: COMMAND ~.
- DEFINE Statement—Define a Logical Name for a File-System Element (see page 309)—New
- ENVAR Statement—Pass Environmental Variables
  - Added "HP Integrity NonStop" to the list of supported job types
  - Added HP Integrity NonStop information to the STDOUT and STDERR operands
  - Added example "Set HP Integrity NonStop Environment Variables"

- FILENAME Statement—Specify File to Monitor (see page 358)—Added syntax for CA WA Agent for HP Integrity NonStop
- PARAMETER Statement—Specify an Input Parameter (HP Integrity NonStop Jobs) (see page 451)—New
- USER Statement—Specify a User ID—Added "HP Integrity NonStop" to the list of job types that require the USER statement.

Contents

Chapter 1: Introduction 21

Overview 21
Agents 22
CA IAS Communication 23
Job Types 24
C-LANG Statements 26
User IDs 28
Passwords 28
Testing for Success 30
Job Output 30
Payload Producing and Payload Consuming Jobs 31
Controlling Jobs 33
Job Classes 33

Chapter 2: Application Services Jobs 35

Application Services Jobs 36
Entity Bean Jobs 37
    Defining Entity Bean Jobs 39
HTTP Jobs 41
    Defining HTTP Jobs 42
JMS Publish and JMS Subscribe Jobs 44
    Defining JMS Publish Jobs 47
    Defining JMS Subscribe Jobs 49
JMX Jobs 51
    Defining JMX-MBean Attribute Get Jobs 53
    Defining JMX-MBean Attribute Set Jobs 54
    Defining JMX-MBean Create Instance Jobs 55
    Defining JMX-MBean Operation Jobs 56
    Defining JMX-MBean Remove Instance Jobs 57
    Defining JMX-MBean Subscribe Jobs 58
POJO Jobs 59
    Defining POJO Jobs 60
RMI Jobs 61
    Defining RMI Jobs 62
Session Bean Jobs 63
    Defining Session Bean Jobs 65

Chapter 3: Database Jobs 67

Database Jobs....67
How Database Trigger Jobs Differ from Database Monitor Jobs....68
Defining Database Monitor Jobs....69
Defining Database Stored Procedure Jobs....71
Supported Data Types....74
Defining Database Trigger Jobs....76
Examples: Monitoring Oracle Database Tables....77
Examples: Monitoring Microsoft SQL Server Database Tables....79
Examples: Monitoring IBM DB2 Database Tables....80
Defining SQL Jobs....81
Examples: Running SQL Statements Against Oracle Database Tables....82
Examples: Running SQL Statements Against Microsoft SQL Server Database Tables....83
Examples: Running SQL Statements Against IBM DB2 Database Tables....85

Chapter 4: File Trigger Jobs 87

File Trigger Jobs....87
Defining File Trigger Jobs....88
Monitor a File that is Owned by a UNIX Owner or Group....90
Monitor a File on a Remote Windows Computer....91

Chapter 5: FTP Jobs 93

FTP Jobs....93
EBCDIC File Transfers....93
Wildcard Characters in File Names....94
Running the Agent as an FTP Client....94
Running the Agent as an FTP Server....95
Defining FTP Jobs....96
Transfer Files Using SSL FTP....99
Compress Data for FTP....101
Send Site-Specific FTP Commands to FTP Servers....102
Verify the FTP Job Status....103

Chapter 6: HP Integrity NonStop Jobs 105

HP Integrity NonStop Jobs....105
Defining HP Integrity NonStop Jobs....105
How to Work with an HP Integrity NonStop Job....106
Specify the Agent Name for an HP Integrity NonStop Job....107
Specify the User ID that an HP Integrity NonStop Job Runs Under....107

Run a Command on the CA WA Agent for HP Integrity NonStop 107
Assign a Logical File Name to a Physical File on an HP Integrity NonStop System 108
Define a Logical Name for a File-System Element on an HP Integrity NonStop System 108
Pass Environment Variables to an HP Integrity NonStop Job 108
Pass a Parameter to an HP Integrity NonStop Job 109
Pass Positional Arguments to an HP Integrity NonStop Job 109

Chapter 7: i5/OS Jobs 111

i5/OS Jobs 111
Running UNIX Workload on a System i5 Computer 112
i5/OS Naming Conventions 112
Defining i5/OS Jobs 113
Pass Positional Parameters 114
Use a User's Library List 114
Pass Keyword Parameters to SBMJOB 114
Set an i5/OS Job's Process Priority 116
Returning a Job's Exit Status to the Scheduling Manager 116
Send a Program's Return Code 117
Send a User-Defined Exit Code 118
Specify Data for a Local Data Area 119

Chapter 8: Monitoring Jobs 121

Monitoring Jobs 121
Defining CPU Monitoring Jobs 122
Defining Disk Monitoring Jobs 123
Defining IP Monitoring Jobs 125
Defining Process Monitoring Jobs 126
Defining Text File Reading and Monitoring Jobs 127
Monitor a File with Data Encoded in a non-ASCII Character Set 129
Defining Windows Event Log Monitoring Jobs 130
Defining Windows Service Monitoring Jobs 132

Chapter 9: Oracle E-Business Suite Jobs 135

Oracle E-Business Suite Jobs 135
Defining Oracle E-Business Suite Copy Jobs 136
Defining Oracle E-Business Suite Request Set Jobs 137
Specify Data for an Individual Program in a Request Set 139
Specify Argument Values to Pass to a Program in a Request Set 141
Defining Oracle E-Business Suite Single Request Jobs 142
Specify Argument Values to Pass to a Program in a Single Request 143

Chapter 10: PeopleSoft Jobs 145

PeopleSoft Jobs 145
Defining PeopleSoft Jobs 146
Distribute a PeopleSoft Report 148
Send the Output of a PeopleSoft Job to a File 149
Send the Output of a PeopleSoft Job to a Printer 150

Chapter 11: SAP Jobs 153

SAP Jobs 153
Defining SAP Batch Input Session Jobs 155
Defining SAP Business Warehouse InfoPackage Jobs 158
Defining SAP Business Warehouse Process Chain Jobs 159
Defining SAP Data Archiving Jobs 160
Defining SAP Event Monitor Jobs 162
Defining SAP Job Copy Jobs 164
Defining SAP Process Monitor Jobs 165
Defining SAP R/3 Jobs 166
Define Recipients for the Job's Output 169
Email the Spool File on Step Completion or Failure 170
Using Success and Failure Messages within an SAP Job 172

Chapter 12: Secure File Transfers 173

Secure Copy and Secure FTP Jobs 173
Defining Secure Copy Jobs 174
Defining Secure FTP Jobs 175

Chapter 13: SNMP Jobs 177

SNMP Jobs 177
Defining SNMP Subscribe Jobs 178
Defining SNMP Trap Send Jobs 179
Defining SNMP Value Get Jobs 181
Defining SNMP Value Set Jobs 183

Chapter 14: UNIX Jobs 185

UNIX Jobs 185
    The Directory the Job Runs Under 186
    Determining Which Shell is Used 187
    Shell Initialization Files 188
Defining UNIX Jobs 190

Verify File Space Before a UNIX Job Starts 191
Pass Positional Arguments in a UNIX Job 191
Pass Environment Variables in a UNIX Job 192
UNIX Environment Variables 193
Define Alternative Input and Output Sources 194
Send a User-Defined Exit Code 195
Specifying the Command or Script Name Without the Full Path in a UNIX Job 196
Specifying the Command or Script Name Using an Environment Variable 197
Set a UNIX Job's Process Priority 197
Running a Script Under a Specific User's Account 198
Modify Resource Limits 199
How to Customize the Job's Runtime Environment 200
Customize the Runtime Environment for a Korn Shell Script 200
Customize the Runtime Environment for a Bourne Shell Script 201
Customize the Runtime Environment for a C Shell Script 201
Run a Perl Script on UNIX 202

Chapter 15: Wake on LAN Jobs 205

Wake on LAN Jobs 205
Defining Wake on LAN Jobs 206

Chapter 16: Web Service Jobs 209

Web Service Jobs 209
Defining Web Service Jobs 211

Chapter 17: Windows Jobs 213

Windows Jobs 213
Defining Windows jobs 214
Verify File Space Before a Windows Job Starts 215
Pass Positional Arguments in a Windows Job 215
Pass Environment Variables in a Windows Job 216
Specifying the Command or Script Name Without the Full Path in a Windows Job 217
Run the Windows Command Interpreter 218
Use a Windows Job Object to Manage Job Processing Properties 218
Access Network Resources 220
Specifying a Password for a User ID 222

Chapter 18: C-LANG Statements 223

Syntax Diagrams 223

ABAPNAME Statement—Specify an ABAP Name 224
ACTION Statement—Specify a Servlet Path 225
AGENT Statement—Specify the Agent Where the Job Runs 226
APPLDISPLNAME Statement—Specify the Display Name of an Oracle Applications Application 227
APPLSHORTNAME Statement—Specify the Short Name of an Oracle Applications Application 228
ARCCLIENT Statement—Specify the SAP Archive Link Client 229
ARCCONNECT Statement—Specify the SAP Archive Link Communication Connection 230
ARCDATE Statement—Specify the SAP Archive Link Archiving Date 231
ARCDOCCLASS Statement—Specify the SAP Archive Link Document Class 232
ARCDOCTYPE Statement—Specify the SAP Archive Document Type 233
ARCFORMAT Statement—Specify the SAP Archive Output Format 234
ARCHOSTLINK Statement—Specify the SAP Archive Link RPC Host 235
ARCINFO Statement—Specify the SAP Archive Link Information 237
ARCMODE Statement—Specify the SAP Archive Mode 238
ARCOBJNAME Statement—Identify Name of SAP Archiving Object 239
ARCOBJTYPE Statement—Specify the Type of SAP External System Archive Object 240
ARCOBJVARIANT Statement—Identify Name of SAP Archiving Object Variant 241
ARCPATH Statement: Specify the SAP Standard Archive Path 242
ARCPRINTER Statement—Specify the SAP Archive Target Printer 243
ARCPROTOCOL Statement—Specify the SAP Archive Storage Connection Protocol 244
ARCREPORT Statement—Specify the SAP Archive Link Report Name 245
ARCSERVICE Statement—Specify the SAP Archive Link RPC Service/Destination 246
ARCSTORAGE Statement—Specify the SAP Archive Link Target Storage System 247
ARCTEXT Statement—Specify the SAP Archive Link Text Information Field 248
ARCUSER Statement—Specify the SAP Archive Data Element for User 249
ARCVERSION Statement—Specify the SAP Archive Version 250
ARGDEFAULTS Statement—Specify Whether to Use Oracle Applications Argument Defaults 251
ARGS Statement—Specify a Parameter Passed to a Stored Procedure (Database Jobs) 253
ARGS Statement—Define Argument Values to Pass (Oracle E-Business Suite Jobs) 255
ARGS Statement—Pass Additional Parameters for the PeopleSoft Report (PeopleSoft Jobs) 256
ARGS Statement—Pass Arguments (UNIX, Windows, and HP Integrity NonStop Jobs) 257
AS400FILE Statement—Identify i5/OS Source File 259
AS400LIB Statement—Specify the Name of a Library 261
ASSIGN Statement—Assign a Logical File Name to a Physical File 262
ATTRIBUTE Statement—Specify the MBean Attribute to Query or Set 264
ATTRIBUTESFILTER Statement—Specify the Attribute Notifications to Subscribe to 265
AUTHORDER Statement—Specify Authentication Protocols 266
AUTHPROTOCOL Statement—Specify the SNMP v3 Authentication Protocol 267
BANNER Statement—Specify Whether to Include an SAP Cover Page 268
BANNERPAGE Statement—Specify Whether to Include an SAP Cover Page with the Report Output 269
BDCERRRATE Statement—Specify the Maximum Acceptable Error Rate 270
BDCEXTLOG Statement—Generate Advanced Logging of the Batch Input Session 271

BDCPROCRATE Statement—Specify the Minimum Acceptable Process Rate 272
BDCSYSTEM Statement—Specify the Name of the Background Server 273
BEAN Statement—Specify the JNDI Name of the Bean 274
BEANNAME Statement—Specify the JNDI Name of the Bean 275
BROADCAST Statement—Specify the Broadcast Address 275
CCEXIT Statement—Specify Type of Exit Code Returned by an i5/OS Job 276
CHAIN Statement—Identify Name of Business Warehouse Process Chain on SAP System 277
CHILDMONITOR Statement—Specify Whether to Monitor the Children of Programs (Oracle E-Business Suite Jobs) 279
CHILDMONITOR Statement—Specify Whether to Monitor Children Jobs (SAP Jobs) 280
CLASSNAME Statement—Specify a Class Name 281
CLPNAME Statement—Identify i5/OS Program to Run 282
CMDNAME Statement—Specify Commands to Run on Windows and UNIX 284
COLUMNS Statement—Specify Line Width of List 288
COMMAND Statement—Specify Commands (HP Integrity NonStop Jobs) 289
COMMAND Statement—Specify Commands to Run (Windows and UNIX Jobs) 290
COMMANDNAME Statement—Specify Commands to Run (Windows and UNIX Jobs) 291
COMMUNITY Statement—Specify the SNMP v1 or v2 Read Community 291
CONNECTION Statement—Specify the JNDI Name of the Connection Factory 292
CONNECTION_FACTORY Statement—Specify the JNDI Name of the Connection Factory 292
CONNECTIONDOMAIN Statement—Specify a Domain for NTLM Connection Authentication 293
CONNECTIONORIGIN Statement—Specify an Origin Host Name for NTLM Connection Authentication 294
CONNECTIONUSER Statement—Specify a User Name for Connection Authentication 295
CONTEXTENGINEID Statement—Specify a Context Engine ID 297
CONTEXTNAME Statement—Specify a v3 Context Name 298
COPIES Statement—Specify the Number of Copies to Print (Oracle E-Business Suite Jobs) 299
COPIES Statement—Specify Number of Copies for SAP Report (SAP Jobs) 299
CPU Statement—Specify Conditions to Monitor CPU Use 299
CREATEMETHOD Statement—Specify a Create Method 302
CREATEPARAMETER Statement—Specify Create Parameters 303
CURLIB Statement—Specify the Name of the Current Library 305
CUSTOMPROP Statement—Specify Value Set Expressions for Default Values 306
DB_URL Statement—Specify Database Resource Location 308
DEFINE Statement—Define a Logical Name for a File-System Element 309
DESC Statement—Describe an Oracle Applications Single Request 312
DESCRIPTION Statement—Describe an Oracle Applications Single Request 312
DEST Statement—Specify an Output Destination File 313
DESTINATION Statement—Specify an Output Destination File 314
DESTINATIONNAME Statement—Specify the JNDI Name of the Topic or Queue 315
DESTNAME Statement—Specify the JNDI Name of the Topic or Queue 316
DIRECTION Statement—Specify the Transfer Direction 318
DISABLE_RESTART Statement—Specify Whether to Disable the Restart Feature for Failed PeopleSoft Jobs 318

DISK Statement—Specify Conditions to Monitor Disk Space 319
DISPLNAME Statement—Specify the Display Name of an Oracle Applications Application 324
DISTRFOLDER Statement—Specify the Name of a PeopleSoft Distribution Detail Folder 324
DISTRLISTROLES Statement—Specify a Distribution List of Roles 325
DISTRLISTUSERS Statement—Specify a Distribution List of Operator IDs 327
DISTRROLES Statement—Specify a Distribution List of Roles 328
DISTRUSERS Statement—Specify a Distribution List of Operator IDs 328
DR Statement—Specify Whether to Disable the Restart Feature for Failed PeopleSoft Jobs 329
EMAILADDR Statement—Specify the Email Addresses on a Distribution List (PeopleSoft Jobs) 329
EMAILADDR Statement—Email Spool File on ABAP Completion or Failure (SAP Jobs) 330
EMAILLOG Statement—Specify Whether to Email PeopleSoft Job Logs 331
EMAILSUBJECT Statement—Define an Email Subject 332
EMAILTEXT Statement—Define the Body Text of an Email 333
EMAILWEBREPORT Statement—Specify Whether to Email a PeopleSoft Web Report 334
ENCODING Statement—Specify the Character Encoding of a File 335
ENDPOINT_URL Statement—Specify a Target Endpoint URL 336
ENGINEID Statement—Specify an Engine ID 337
ENVAR Statement—Pass Environment Variables 338
EVENT Statement—Specify the SAP Event to Monitor or Trigger 345
EVENTCATEGORY Statement—Specify a Windows Event Category 347
EVENTCOMPUTER Statement—Specify a Local Computer for Windows Event Log 348
EVENTDESCRIPTION Statement—Specify a Windows Event Description 349
EVENTID Statement—Specify the Windows Event IDs to Monitor 350
EVENTLOG Statement—Specify the Name of a Windows Event Log 351
EVENTSOURCE Statement—Specify a Windows Event Source 352
EVENTTIME Statement—Specify the Date and Time of a Windows Event Log 353
EVENTTYPE Statement—Specify a Windows Event Type 354
EXITCODE Statement—Identify Success or Failure by Exit Code 355
EXPIRATION Statement—Specify the Number of Days Before the Spool Request is Deleted 357
FAILUREMSG Statement—Specify Failure Message for ABAP 358
FILENAME Statement—Specify File to Monitor 358
FILESYSTEM Statement—Verify File Space Required to Start a Job 374
FILTER Statement—Specify a Filter Using Regular Expression Logic (Application Services/Web Service Jobs) 375
FILTER Statement—Specify a Filter Using Regular Expression Logic (SNMP Jobs) 377
FINDERMETHOD Statement—Specify a Finder Method 378
FINDERPARAMETER Statement—Specify Finder Parameters 379
FTPFORMAT Statement—Specify the FTP Format 381
HOST Statement—Specify Database Resource Location 381
INFOPACK Statement—Identify Name of Business Warehouse InfoPackage on SAP System 381
INITIAL_CONTEXT Statement—Specify an Initial Context Factory 382
INITIAL_CONTEXT_FACTORY Statement—Specify an Initial Context Factory 383
INTERACTIVE Statement—Specify Whether to Run a Windows Job in Interactive Mode 384

INVOCATIONTYPE Statement—Specify the HTTP Method Type 385
IPADDRESS Statement—Specify the IP Address to Monitor 386
IPPORT Statement—Specify the Port Number at the IP Address to Monitor 386
JNDIUSER Statement—Specify a JNDI User ID 387
JOB_CRITERIA Statement—Specify the Evaluation Criteria for a Return String 389
JOBCLASS Statement—Assign a Job Class to a Job 392
JOBCOPY Statement—Copy an Existing SAP Job 394
JOBCRIT Statement—Specify the Evaluation Criteria for a Return String 395
JOBD Statement—Specify the Job Description 396
JOBNAME Statement—Specify the Job Name (i5/OS Jobs) 397
JOBNAME Statement—Specify Job Name (SAP Jobs) 397
JOBOBJECT Statement—Associate Windows Job with a Windows Job Object 398
JOBQ Statement—Specify the Job Queue for a Program 401
JUSER Statement—Specify a JNDI User ID 402
LANGUAGE Statement—Specify the Language the ABAP Uses 402
LDA Statement—Specify the Data for the Local Data Area 403
LIBL Statement—Specify the Libraries for an I5/OS Job 404
LINES Statement—Specify the Number of Lines per List Page 406
LOCALFILENAME Statement—Specify Local Filenames (FTP Job) 407
LOCALNAME Statement—Specify Local Filenames (FTP Jobs) 409
LOCALNAME Statement—Specify the Local File to Transfer (Secure Copy and Secure FTP Jobs) 410
LOCALUSER Statement—Specify a User ID on the Agent Computer for the Downloaded File 412
LOCATION Statement—Specify a Service Provider URL 413
LOGNAME Statement—Specify Name of Windows Event Log 415
MAC Statement—Specify the Media Access Control (MAC) Address 416
MBEAN Statement—Specify the Name of the MBean 417
MESSAGECLASS Statement—Specify the Java Class of the JMS Message 418
METHOD Statement—Specify the Method to be Invoked Remotely 419
METHODNAME Statement—Specify the Method to be Invoked Remotely 420
MIB Statement—Specify a MIB File Name 421
MODIFYPARAMETER Statement—Specify Modify Parameters 423
MON_COND Statement—Specify a Condition to Monitor 425
MON_TYPE Statement—Specify the Type of Database Change to Monitor 426
MONCOND Statement—Specify a Condition to Monitor 426
MONITORDELAY Statement—Define the Time to Wait Before Checking the Status of Children Programs 427
MONTYPE Statement—Specify the Type of Database Change to Monitor 428
MSGCLASS Statement—Specify the Java Class of the JMS Message 429
NOTIFYDUSERS Statement—Specify Users to Notify Using Display Names 430
NOTIFYUSERS Statement—Specify Users to Notify Using Short Names 431
OAUSER Statement—Specify an Oracle Applications User Name 432
OBJNAME Statement—Identify Name of SAP Archiving Object 433
OPERATION Statement—Specify the Operation to be Invoked 434

OPERATIONTYPE Statement—Specify the Operation Type 435
OTHERS Statement—Pass Keyword Parameters to SBMJOB 436
OUTDESTFORMAT Statement—Specify the Output Format for a PeopleSoft Report 437
OUTDESTPATH Statement—Specify an Output Path for a PeopleSoft Job 439
OUTDESTTYPE Statement—Specify the Output Type for a PeopleSoft Report 441
OUTFILE Statement—Specify File for Results of SQL Query 443
OUTPUT_FILE Statement—Specify File for Results of SQL Query 443
OUTPUTFORMAT Statement—Specify Output Format for a Single Request or Request Set 444
PARAM Statement—Specify Positional Parameters (i5/OS Jobs) 445
PARAM Statement—Specify Data Selection Criteria (SAP Jobs) 446
PARAMETER Statement—Specify Input Parameters (Application Services Jobs) 448
PARAMETER Statement—Specify an Input Parameter (HP Integrity NonStop Jobs) 451
PARAMETER Statement—Specify Input Parameters (SNMP Jobs) 453
PARAMETER Statement—Specify Input Parameters (Web Service Jobs) 456
PORTNAME Statement—Specify the Port Name Within the Namespace 459
PRINTCOPIES Statement—Specify the Number of Copies to Print (Oracle E-Business Suite Jobs) 460
PRINTCOPIES Statement—Specify the Number of Print Copies (SAP Jobs) 462
PRINTCOVER Statement—Specify the SAP Cover Page Text 463
PRINTDATASET Statement—Specify Data Set Name for SAP Print Spool 464
PRINTDEPARTMENT Statement—Specify Department Name 465
PRINTDEST Statement—Specify Print Destination 466
PRINTER Statement—Specify a Printer 467
PRINTFOOTER Statement—Specify Whether to Print a Footer on an SAP Report 468
PRINTFORMAT Statement—Specify an Output Format for an SAP Report 469
PRINTHOSTPAGE Statement—Print Host Page 470
PRINTIMMED Statement—Specify Immediate Print After Job Completion 471
PRINTNEWSPOOL Statement—Create New Spool Request 472
PRINTPRIORITY Statement—Specify the Priority of the SAP Spool Request 473
PRINTPW Statement—Specify SAP Authorization String for Viewing Print Spool List 474
PRINTREL Statement—Specify Deletion of Spool Request 475
PRINTREQTYPE Statement—Specify Print Request Type 476
PRINTSPOOLNAME Statement—Specify Name of SAP Print Spool 477
PRINTSTYLE Statement—Specify an Oracle Applications Print Style 478
PRIVPROTOCOL Statement—Specify the SNMP v3 Privacy Protocol 479
PROCESS Statement—Specify the Process to be Monitored (Process Monitoring Jobs) 480
PROCESS Statement—Specify the Process Status to Monitor (SAP Jobs) 483
PROCESS_NAME Statement—Specify a PeopleSoft Process to Run 483
PROCESS_PRIORITY Statement—Specify the Process Priority of a UNIX or i5/OS Job 484
PROCESS_TYPE Statement—Specify the Type of PeopleSoft Process to Run 485
PROCESSCHAIN Statement—Identify Name of Business Warehouse Process Chain on SAP System 485
PROCESSMONITOR Statement—Specify the SAP Process Status to Monitor 486
PROCESSNAME Statement—Specify a PeopleSoft Process to Run 488

PROCESSTYPE Statement—Specify the Type of PeopleSoft Process to Run 489
PROGARGS Statement—Define Argument Values to Pass to a Request Set Program 490
PROGCOPIES Statement—Specify the Number of Copies to Print for a Request Set Program 491
PROGDATA Statement—Identify an Oracle Applications Program 492
PROGNOTIFYDUSERS Statement—Specify Users to Notify Using Displays Names 494
PROGNOTIFYUSERS Statement—Specify Users to Notify Using Short Names 496
PROGOUTPUTFORMAT Statement—Specify the Output Format for a Program in a Request Set 497
PROGPRINTCOPIES Statement—Specify the Number of Copies to Print for a Request Set Program 498
PROGPRINTER Statement—Specify a Printer for a Request Set Program 500
PROGPRINTSTYLE Statement—Specify Print Style for Request Set Program 501
PROGQUOTEDEFAULT Statement—Specify Whether to Quote Resolved Expressions in Default Values 503
PROGRAM Statement—Specify the Short Name of an Oracle Applications Single Request Program 504
PROGRAMDATA Statement—Identify an Oracle Applications Program 505
PROGRAMDISPL Statement—Specify the Display Name of an Oracle Applications Single Request Program 505
PROGSAVEOUTPUT Statement—Specify Whether to Save Output from a Request Set Program 506
PROGTEMPLATELANG Statement—Specify the Template Language for a Program in a Request Set 507
PROGTEMPLATETERR Statement—Specify the Template Territory for a Program in a Request Set 508
PROVIDER_URL Statement—Specify a Service Provider URL 509
PROXYDOMAIN Statement—Specify a Domain for Proxy Authentication 510
PROXYHOST Statement—Specify a Proxy Host 511
PROXYORIGIN Statement—Specify an Origin Host Name for Proxy Authentication 512
PROXYPORT Statement—Specify a Proxy Port 514
PROXYUSER Statement—Specify a User for Proxy Authentication 515
PSOPRID Statement—Specify Operator ID for a PeopleSoft Report 517
PUTINOUTBOX Statement—Specify Whether to Save Outgoing Documents to the SAPoffice Outbox 518
QUOTEDEFAULT Statement—Specify Whether to Quote Resolved Expressions in Default Values 519
RANGE Statement—Specify Search Range 520
RECIPIENT Statement—Specify Recipient Name for SAP Cover Page 521
RECIPIENTBCC Statement—Specify Whether to Send a Blind Carbon Copy of the SAP Spool 522
RECIPIENTCC Statement—Specify Whether to Send a Carbon Copy of the SAP Spool 523
RECIPIENTEXPRESS Statement—Specify Whether to Send the SAP Spool Using Express Delivery 524
RECIPIENTFORWARD Statement—Allow Forwarding of SAP Spool 525
RECIPIENTPRINT Statement—Specify Whether to Allow Printing of the SAP Spool 526
RECIPIENTTYPE Statement—Specify the Recipient Type for the SAP Spool 527
REMOTECLASSNAME Statement—Specify a Remote Class Name 528
REMOTEDIR Statement—Specify a Remote Directory 529
REMOTEFILENAME Statement—Specify Remote Filename (FTP Job) 530
REMOTENAME Statement—Specify Remote Filenames (FTP Jobs) 533
REMOTENAME Statement—Specify a Remote File Name (Secure Copy and Secure FTP Jobs) 534
REQSET Statement—Specify the Short Name of an Oracle Applications Request Set 535
REQSETDISPL Statement—Specify the Display Name of an Oracle Applications Request Set 536
REQUESTID Statement—Specify Request ID of the Oracle Applications Request to Copy 537

REQUESTSET Statement—Specify the Short Name of an Oracle Applications Request Set 538
RESPNAME Statement—Specify an Oracle Applications Responsibility Name 539
RETURN Statement—Specify JDBC Data Type for a Return Value 540
RETURN_DATA_TYPE Statement—Specify JDBC Data Type for a Return Value 540
RETURNCLASS Statement—Specify a Return Class Name 542
RETURNNAMESPACE Statement—Specify an XML Namespace 543
RETURNXML Statement—Specify an XML Type 544
RFCDEST Statement—Specify SAP Destination 546
RUNCONTROLARGS Statement—Specify the Arguments for a PeopleSoft Run Control Table 547
RUNCONTROLID Statement—Specify PeopleSoft Run Parameters 549
RUNCONTROLTABLE Statement—Specify the Table that Contains the PeopleSoft Run Parameters 550
SAPCLIENT Statement—Specify SAP Client 552
SAPEMAILADDR Statement—Email SAP Spool File on Job Completion or Failure 553
SAPFAILUREMSG Statement—Specify Failure Message for SAP Job 554
SAPJOBCLASS Statement—Specify SAP Job Class 555
SAPJOBNAME Statement—Specify SAP Job Name 556
SAPLANGUAGE Statement—Specify SAP Language for Login 557
SAPSUCCESSMSG Statement—Specify Success Message for SAP Job 558
SAPTARGETSYSTEM Statement—Specify SAP Application Server 559
SAPUSER Statement—Specify the SAP System User the Job Runs Under 560
SAVEOUTPUT Statement—Specify Whether to Save Output from an Oracle E-Business Suite Job 562
SCRIPTNAME Statement—Specify a UNIX Shell Script to Run 563
SEARCHRANGE Statement—Specify the Range to be Searched 566
SERVERADDR Statement—Specify a Remote Server Name 572
SERVERNAME Statement—Specify a Server to Run the PeopleSoft Job 573
SERVERPORT Statement—Specify a Remote Server Port 575
SERVICE Statement—Specify Windows Service Name 576
SERVICENAME Statement—Specify the Web Service Name Within the Target Namespace (Web Service Jobs) 577
SERVICENAME Statement—Specify the Name of the Windows Service to be Monitored (Windows Service Monitoring Jobs) 578
SERVLET_URL Statement—Specify a Host URL 579
SHELL Statement—Specify UNIX Shell to Run Script 582
SHORTNAME Statement—Specify the Short Name of an Oracle Applications Application 583
SITECMD Statement—Specify Site-specific FTP Commands 583
SKIPPARMUPDATES Statement—Specify Whether to Update Job Parameters 584
SNMPNODE Statement—Specify an SNMP OID, Host, Port, Version, and Other Options 586
SNMPUSER Statement—Specify an SNMP v3 User Name 589
SPNAME Statement—Specify Stored Procedure or Stored Function to Run 590
SPOOLRECIPIENT Statement—Specify SAP Spool Recipient 590
SQL Statement—Specify SQL to Run Against a Database Table 591
SQL_COMMAND Statement—Specify SQL to Run Against a Database Table 592

STARTMODE Statement—Specify Startmode for Readied Job 592
STATUS Statement—Specify the Status of the Process, Service, or IP Address to Monitor 593
STEPUSER Statement—Specify the SAP System User the ABAP Runs Under 596
STORED_PROCEDURE Statement—Specify Stored Procedure or Stored Function to Run 597
SUCCESSMSG Statement—Specify Success Message for ABAP 598
TABLE_NAME Statement—Specify the Name of the Table to Monitor 599
TABLENAME Statement—Specify the Name of the Table to Monitor 601
TARGETNAMESPACE Statement—Specify a Target Namespace 601
TARGETOSTYPE Statement—Specify the Remote Operating System Type 602
TEMPLATELANG Statement—Specify the Template Language for a Single Request or Request Set 604
TEMPLATETERR Statement—Specify the Template Territory for a Single Request or Request Set 605
TEXTFILE Statement—Specify a Text File Name and Location 606
TEXTSTRING Statement—Specify a Text String to Search For 609
TIMEFORMAT Statement—Define a Time Format 613
TIMEOUT Statement—Specify the Ping Timeout 616
TIMEZONE Statement—Specify a Time Zone 617
TOPIC Statement—Specify Whether to Publish or Subscribe to a Topic or Queue 618
TRANSFERCODETYPE Statement—Specify the FTP Format 619
TRANSFERDIRECTION Statement—Specify the File Transfer Direction 625
TRIG_COND Statement—Specify a Condition to Monitor 626
TRIG_TYPE Statement—Specify the Type of Database Change to Monitor For 628
TRIGCOND Statement—Specify a Condition to Monitor 629
TRIGTYPE Statement—Specify the Type of Database Change to Monitor 629
TYPESFILTER Statement—Specify the Notification Types to Subscribe to 630
ULIMIT Statement—Specify UNIX Resource Limits 631
URL Statement—Specify the URL of the JMX Server 633
USER Statement—Specify a User ID 634
USER_TYPE Statement—Specify the Oracle Database User Type 638
USERTYPE Statement—Specify the Oracle Database User Type 639
USESETDEFAULTS Statement—Specify Whether Request Set Defaults Take Precedence over Program Defaults 640
VARIANT Statement—Specify Variant Name 641
WAITMODE Statement—Specify Whether to Monitor Conditions Immediately or Continuously 642
WAKEHOST Statement—Specify the Computer to Ping 643
WAKEPASSWORD Statement—Specify the Wake on LAN Password 644
WAKEPORTS Statement—Specify the Ports to Ping 645
WEBPOSTING Statement—Specify Whether to Post the SAP Job Log and Spool on the Web 646
WSDL_URL Statement—Specify a WSDL URL 647

Chapter 1: Introduction

This section contains the following topics:

Overview (see page 21)
Agents (see page 22)
CA IAS Communication (see page 23)
Job Types (see page 24)
C-LANG Statements (see page 26)
User IDs (see page 28)
Passwords (see page 28)
Testing for Success (see page 30)
Job Output (see page 30)
Payload Producing and Payload Consuming Jobs (see page 31)
Controlling Jobs (see page 33)
Job Classes (see page 33)

Overview

CA Integrated Agent Services (CA IAS) lets CA mainframe scheduling managers (CA Workload Automation SE, CA Scheduler Job Management, and CA Jobtrac Job Management) submit and track work to various distributed platforms such as Windows and UNIX. CA IAS also lets the scheduling managers monitor CPU and disk usage on distributed platforms, transfer files, and more.

CA IAS provides a communication layer between the CA scheduling manager and CA Workload Automation agents.

The CA mainframe scheduling manager is responsible for determining when jobs are sent to the agent.

Each scheduling manager has its own method for defining jobs and when they are executed. The scheduling managers also have their own methods for interacting with CA IAS. For more information about how the scheduling manager interacts with CA IAS, see the documentation for your scheduling manager.

This document provides details that are common to all three scheduling managers.

Agents

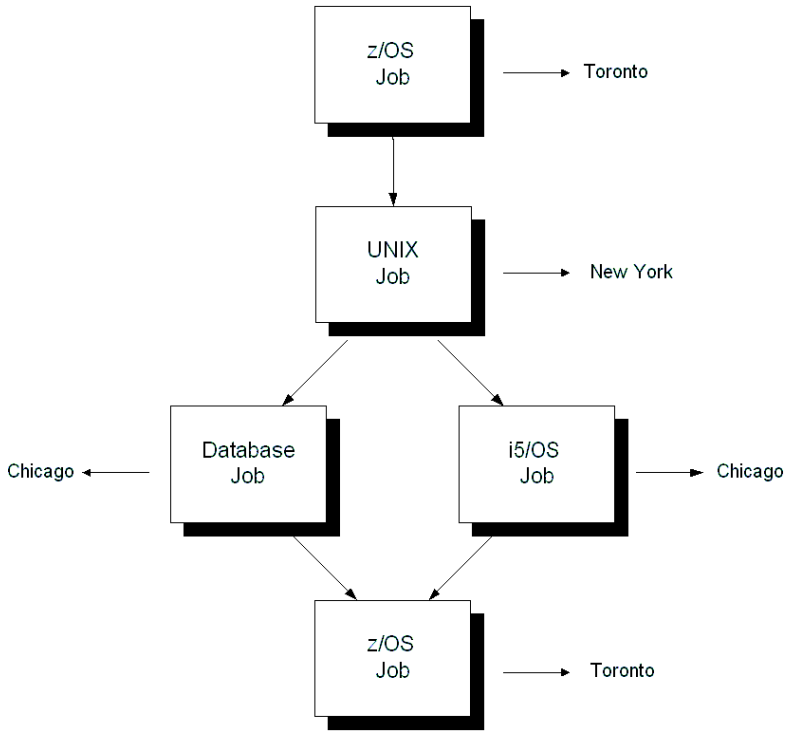
Agents are the key integration components of CA's workload automation products. Agents let you automate, monitor, and manage workload on all major platforms, applications, and databases. To run workload on a particular system, you need to install an agent on that system. For example if your workload must run on a UNIX computer, you can install and configure the CA WA Agent for UNIX to run UNIX scripts, execute UNIX commands, transfer files using FTP, monitor file activity on the agent computer, and perform many other tasks.

You can extend the functionality of the agent by installing one or more agent plug-ins into the agent installation directory. If you have a relational database such as Oracle, for example, you can install a database agent plug-in to query and monitor the database. Other agent plug-ins are also available, including Application Services, Oracle E-Business Suite, PeopleSoft, SAP, and Web Services. For more information, refer to the appropriate *Implementation Guide* for each agent plug-in.

Note: The agent plug-ins are only available for UNIX and Windows operating environments.

Example: Run Workload with Different Types of Jobs

The following workload contains z/OS jobs, a UNIX job, an database job, and an i5/OS job, running on different computers, in different locations:



CA IAS Communication

CA IAS provides the communication between the CA mainframe scheduling manager and the agent.

The scheduling manager determines when a job is eligible to be executed. In general, CA IAS (agent) jobs are treated like normal z/OS jobs with predecessors, early start times, and so on. For more information, see the documentation for your scheduling manager.

When the scheduling manager determines that the job can be executed, it sends a request to CA IAS. CA IAS formats the request into a message format that the agent understands and then transmits the message to the agent to process. If the agent is not immediately available, either because the system is down or unavailable due to network issues, CA IAS then stores the message. The message is sent automatically to the agent when the agent becomes available.

Users can also issue commands to CA IAS through the scheduling manager. After the user issues the command, CA IAS sends the command to the agent to process. After the agent processes the command, it sends the results back to CA IAS. CA IAS then sends the results of the command to the scheduling manager, which displays the results for the user.

Job Types

The agents can perform many different categories of work. For example, a job can run a command file on a Windows system, perform a database query, or execute a defined job in PeopleSoft.

Each category of work requires a unique set of information to process the request properly. For example, the Windows job only requires the name of the script to execute, while the SQL job requires a database name and location and an SQL query to run.

Each category of work is assigned a job type. Job types define the type of work to be scheduled. For example, you can create an NT_JOB job to run a Windows command file, an SQL_JOB job to perform a database query, or a PS_JOB job to perform work in PeopleSoft. Each scheduling manager has its own method for assigning a job type to jobs.

Note: Each job type requires one or more agents. For more information about a job type's required statements and agents, see the description for that job type.

CA IAS supports the following job types:

Job Type	Full Name
AS400_JOB	i5/OS
BDC_JOB	SAP Batch Input Session
BWIP_JOB	SAP Business Warehouse InfoPackage
BWPC_JOB	SAP Business Warehouse Process Chain
CPU_MON	CPU Monitoring
DB_MON	Database Monitor
DB_TRIG	Database Trigger
DBSP_JOB	Database Stored Procedure
DISK_MON	Disk Monitoring
EJB_JOB	Session Bean
EJBE_JOB	Entity Bean
EVENTLOG_MON	Windows Event Log Monitoring
FILE_TRIGGER	File Trigger
FTP_JOB	FTP
HTTP_JOB	HTTP
IP_MON	IP Monitoring

Job Type	Full Name
JMSP_JOB	JMS Publish
JMSS_JOB	JMS Subscribe
JMXA_JOB	JMX-MBean Attribute Set
JMXB_JOB	JMX-MBean Attribute Get
JMXN_JOB	JMX-MBean Create Instance
JMXO_JOB	JMX-MBean Operation
JMXR_JOB	JMX-MBean Remove Instance
JMXS_JOB	JMX-MBean Subscribe
MF_JOB	Micro Focus
OA_JOB	Oracle E-Business Suite Single Request or Request Set
OAC_JOB	Oracle E-Business Suite Copy Job
POJO_JOB	POJO
PROCESS_MON	Process Monitoring
PS_JOB	PeopleSoft
RMI_JOB	RMI
SAP_JOB	SAP R/3 or Job Copy
SAPA_JOB	SAP Data Archiving
SAPE_JOB	SAP Event Monitor
SAPM_JOB	SAP Process Monitor
SCP_JOB	Secure Copy
SERVICE_MON	Windows Service Monitoring
SFTP_JOB	Secure FTP
SNPC_JOB	SNMP Subscribe
SNPE_JOB	SNMP Trap Send
SNPG_JOB	SNMP Value Get
SNPS_JOB	SNMP Value Set
SQL_JOB	SQL
NONSTOP_JOB	HP Integrity NonStop Job
TEXT_MON	Text File Reading and Monitoring
NT_JOB	Windows

Job Type	Full Name
UNIX_JOB	UNIX
WEB_SERV	Web Service
WOL_JOB	Wake on LAN

C-LANG Statements

The information required to process jobs is passed to CA IAS using C-LANG (control language) statements. C-LANG is sometimes referred to as CLANG.

Each job type has one or more required C-LANG statements and can also have one or more optional C-LANG statements. For example, the Windows job type (NT_JOB) requires the CMDNAME statement to specify the name of the command name you want to run. Because a set of C-LANG statements make up a job definition, we also refer to these statements as job definition statements.

C-LANG statements are typically stored in PDS members, usually one member per job. The specific dataset that C-LANG statements are stored in depends on the scheduling manager. The dataset must be defined as fixed-block (FB) with a record length (LRECL) of 80.

The following syntax rules apply to C-LANG:

- C-LANG statements can start in any column from 1 to 72.
- C-LANG statements can be continued onto the next line by coding a plus sign (+) or minus sign (-) as the last non-blank character on a line. If + is used, any leading blanks on the next line are removed from the statement. If - is used, any leading blanks on the next line are kept in the statement.
- As a general rule, if duplicate C-LANG statements are entered, the value for the last one entered is passed to the agent. Exceptions to this rule (for example, the ARGS statement for Oracle E-Business Suite) are documented.
- The data CA IAS sends to agents is case-sensitive. In general, the values of operands (parameters) in C-LANG statements are case-sensitive and must be entered in the proper case.
- C-LANG statement names (for example, CMDNAME or ARGS) are not case sensitive. They can be entered in upper, lower, or mixed case. By convention, this guide uses uppercase.
- Comments can be included by coding /* at the beginning of the comment and */ at the end of the comment.

Example: Using the Plus Sign as a Continuation Character

The following example uses the plus sign (+) as a continuation character:

```
CMDNAME ‘C:\Program Files+
     \MyDir\script.bat'
```

Because leading blanks on the next line are removed with +, it is equivalent to the following statement:

```
CMDNAME ‘C:\Program Files\MyDir\script.bat’
```

Example: Using the Minus Sign as a Continuation Character

The following example uses the minus sign (-) as a continuation character:

```
ARGS Parm1-
  Parm2
```

Because leading blanks on the next line are kept with -, it is equivalent to the following statement:

```
ARGS Parm1  Parm2
```

Note: In this statement, two blanks separate Parm1 and Parm2 because Parm2 follows two leading spaces in the preceding continuation statement.

Example: Using a Plus Sign Preceded by as Space as a Continuation Character

The following example uses a plus sign (+) preceded by a space as a continuation character:

```
ARGS Parm1 +
  Parm2
```

Because leading blanks on the next line are removed with +, it is equivalent to the following statement;

```
ARGS Parm1 Parm2
```

Note: In this statement, only one blank separates Parm1 and Parm2. The blank is taken from the space before the plus sign in the preceding continuation statement; the leading blanks before Parm2 have been removed.

User IDs

Most CA IAS job types let you specify a user ID under whose authority the job runs. The user ID is specified in a C-LANG statement such as USER.

Several CA IAS job types do not accept the USER statement, but accept another similar C-LANG statement to specify the user ID. For example, the Oracle E-Business Suite Single Request or Request Set job type (OA_JOB) uses the OAUSER statement instead of the USER statement.

Note: The scheduling manager can provide a method to include a user ID automatically when submitting a job, removing the need to code the user ID in a C-LANG statement.

Passwords

Most platforms and shops require a password when logging on a user ID. CA IAS provides a mechanism for storing encrypted passwords and retrieving them during job submission.

Storing passwords in CA IAS can minimize the effort involved when passwords expire and must be changed. For example, a password can be stored once for all jobs using a specific user ID on a given agent. When the password must be changed, it can be changed in one place. All jobs using that user ID on that agent immediately start using the new password.

Each scheduling manager has its own interface to CA IAS, including the facility for defining passwords. See the documentation for your scheduling manager for more information about defining passwords.

Passwords are *never* displayed. Passwords are encrypted before being stored or transmitted.

A password is associated with a user ID and any combination of agent name, job type, and namespace (such as a domain). Passwords are searched in a most-specific to least-specific order. The order of precedence is agent name, followed by job type, followed by namespace.

Example: How Passwords are Matched with User IDs

Suppose the following passwords have been defined:

User ID	Agent Name	Job Type	Namespace	Password
USERA				ONE
USERA	SYSTEMA			TWO

User ID	Agent Name	Job Type	Namespace	Password
USERA		NT_JOB		THREE
USERA	SYSTEMB	NT_JOB		FOUR
USERA			DOMAINX	FIVE

Consider the following scenarios:

- A job with job type NT_JOB runs on system SYSTEMB with a user ID of USERA.

  In this scenario, password FOUR gets used because it is the best match.

- A job with job type NT_JOB runs on system SYSTEMC with a user ID of USERA.

  In this scenario, password THREE gets used because it is the best match.

- A job with job type NT_JOB runs on system SYSTEMA with a user ID of USERA.

  In this scenario, password TWO gets used because agent name is searched before job type.

- A job with job type UNIX_JOB runs on system SYSTEMX with a user ID of USERA.

  In this scenario, password ONE gets used because it is the only possible match.

Note: The namespace only applies to selected job types. For example, an HTTP job (HTTP_JOB) accepts two user IDs: a connection user ID and a proxy user ID. HTTP jobs use domain names to distinguish between two user IDs that have the same name, but different passwords. The namespace lets you store multiple passwords for the same user ID/agent/job type combination. Individual job types that take advantage of this feature are described in detail later in this guide.

Testing for Success

When the agent starts a job, it sends a message to notify CA IAS that the job is now "executing". CA IAS passes the information to the scheduling manager so that the job can be marked as “started”.

When the job completes, the agent sends another message to notify CA IAS that the job is now "complete" or has "failed".

Traditionally, scheduling managers use a job's return code to determine whether the job completed successfully or failed. The scheduling manager determines the success of each job after it has completed.

Agent jobs run through CA IAS are evaluated differently. The agent determines the success or failure of each job and notifies the scheduling manager.

Most job types let you code one or more EXITCODE statement to specify acceptable return codes for the job. These EXITCODE statements control how the agent determines the success or failure of the job. By default, an exit code of 0 (zero) indicates job success and any other code indicates job failure. You can use the EXITCODE statement to indicate an exit code other than 0 or a range of codes as job success.

Job Output

Most job types produce output that the agent stores in a spool file. Depending on the scheduling manager, you can view the spool file through commands, screens, or a graphical user interface. For more information about viewing the spool file, see the documentation for your scheduling manager.

During and after job execution, the agent can return one or more “status messages” about the job. The scheduling manager displays the status information in its regular workload displays or through other methods.

Some job types return additional information in the form of variables and values. You can use these output variables as input to a subsequent job. For example, the result of an SQL query could be stored as a variable and then passed to a job or used as a trigger condition.

Note: Not all scheduling managers support the passing of output as variables.

Payload Producing and Payload Consuming Jobs

A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer.

The following job types are payload producing jobs:

- JMS Subscribe

  Note: The appservices.jms.subscribe.persist parameter must be set to true in the agent's agentparm.txt file for JMS Subscribe jobs to be payload producing jobs.
- JMX-MBean Attribute Get
- JMX-MBean Attribute Set
- JMX-MBean Operation
- POJO
- RMI
- Session Bean
- SNMP Subscribe
- SNMP Trap Send (for INFORM messages only)
- SNMP Value Get
- SNMP Value Set
- Web Service

The serialized Java object produced by a payload producing job is stored on the agent computer in the spool directory, using the job name and a numeric suffix as the file name, or in a destination file you specify.

A payload consuming job is a job that uses the output from a payload producing job as a parameter's input value.

The following job types are payload consuming jobs:

- Entity Bean
- JMS Publish
- JMX-MBean Attribute Set
- JMX-MBean Create Instance
- JMX-MBean Operation
- POJO
- RMI
- Session Bean

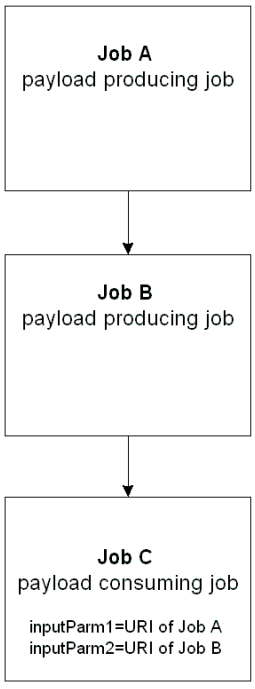
- SNMP Trap Send
- SNMP Value Set
- Web Service

A URI (uniform resource identifier) stores the location of the binary output produced by a payload producing job. From your scheduling manager, you can retrieve the value of the URI from a payload producing job and pass the URI as input to a payload consuming job. When the payload consuming job runs, it takes the binary output referenced by the URI as an input parameter.

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Example: Pass Payload Producing Job Output as Input to Another Job

The following diagram shows the relationship between three jobs in a job flow. Job A and Job B are payload producing jobs that produce binary output. Job C is a payload consuming job that takes two parameters, inputParm1 and inputParm2. Job C uses the output from Job A and Job B as input values. In the definition of Job C, the value of inputParm1 is specified as the URI of Job A and the value of inputParm2 is specified as the URI of Job B.



Controlling Jobs

Some job types can be canceled while executing or held (and released) after being submitted but before starting. Depending on the scheduling manager, you can use commands, screens, or a graphical user interface to control the jobs. For more information about controlling jobs, see the documentation for your scheduling manager.

Job Classes

Most job types support job classes. Job classes are defined on the agent and can be used to limit the number of various types of work the agent can run at one time. For example, an agent might be able to run up to 100 PAYROLL jobs at one time, but only run up to 20 BACKUP jobs at one time. To allow the correct number of jobs to run at one time, the agent administrator must define separate job classes (PAYROLL and BACKUP) on the agent. For more information about defining initiators and classes, see the agent documentation.

The JOBCLASS C-LANG statement associates job classes with jobs. The following statements specify the job classes for the payroll and backup applications, respectively:

```
JOBCLASS  PAYROLL
JOBCLASS  BACKUP
```

Chapter 2: Application Services Jobs

This section contains the following topics:

Application Services Jobs (see page 36)
Entity Bean Jobs (see page 37)
HTTP Jobs (see page 41)
JMS Publish and JMS Subscribe Jobs (see page 44)
JMX Jobs (see page 51)
POJO Jobs (see page 59)
RMI Jobs (see page 61)
Session Bean Jobs (see page 63)

Application Services Jobs

Application Services jobs let you manage entity beans, session beans, and MBeans, publish and consume JMS messages, invoke programs over HTTP, and run other types of Java-based workload.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

You can define the following Application Services jobs:

Entity Bean (see page 37)

Lets you create an entity bean, update the property values of an existing entity bean, or remove an entity bean from the database.

HTTP (see page 41)

Lets you invoke a program over HTTP or HTTPS in a similar way to a web browser. For example, you can use the HTTP job to invoke a CGI script, a Perl script, or a servlet. The HTTP job sends a URL over HTTP using the GET method or a form over HTTP using the POST method.

JMS Publish (see page 44)

Lets you send a message to a queue or publish a message to a topic on a JMS server.

JMS Subscribe (see page 44)

Lets you consume messages from a queue or topic on a JMS server.

JMX-MBean Attribute Get (see page 51)

Lets you query a JMX server for the value of an MBean attribute. The returned value is stored on the computer where the Application Services agent plug-in resides.

JMX-MBean Attribute Set (see page 51)

Lets you change the value of an MBean attribute on a JMX server.

JMX-MBean Create Instance (see page 51)

Lets you create an MBean on a JMX server.

JMX-MBean Operation (see page 51)

Lets you invoke an operation on an MBean on a JMX server.

JMX-MBean Remove Instance (see page 51)

Lets you remove an MBean from a JMX server.

JMX-MBean Subscribe (see page 51)

Lets you monitor an MBean for a single notification or monitor continuously for notifications.

POJO (see page 59)

Lets you instantiate a class to create a Java object and invoke a method on it. The job is restricted to classes that take constructors with no arguments (default constructors). You can use the POJO job to invoke custom Java code on a local computer.

RMI (see page 61)

Lets you set up interaction between Java objects on different computers in a distributed network. Using an RMI job, you can access a remote server and can invoke a method on a Java object.

Session Bean (see page 65)

Lets you access a session bean on an application server. This job type can make a Remote Procedure Call (RPC) to the session bean, invoke a method that defines the business logic, pass parameters to the method, and have the results returned as serialized Java output. You can access stateless and stateful session beans using the Session Bean job.

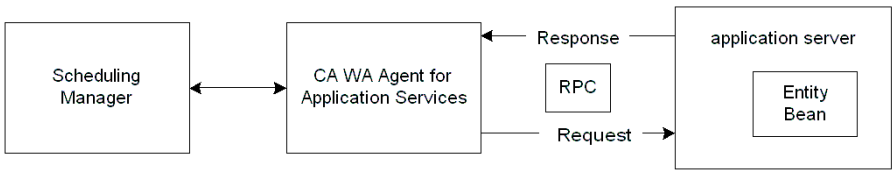
Entity Bean Jobs

An entity bean represents a data object, such as a customer, an order, or a product. Entity beans may be stored in a relational database, where each instance of the bean corresponds to a row in a database table. Each entity bean has a unique identifier known as a primary key, which is used to find a specific instance of the bean within the database. For example, a customer entity bean may use the customer number as its primary key.

Unlike session beans, which are destroyed after use, entity beans are persistent. You can use an entity bean under the following conditions:

- The bean represents a business entity, not a procedure. For example, you use an entity bean to represent an order and use a session bean to represent the procedure to process the order.
- The state of the bean must be stored. For example, if the bean instance terminates or the application server shuts down, the bean's state will still exist in a database.

The following diagram shows the functional relationship between the scheduling manager, CA WA Agent for Application Services, and an entity bean residing on an application server:



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. To find the entity bean, the agent uses the bean's Java Naming and Directory Interface (JNDI) name along with its finder method.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

To define an Entity Bean job, you require the following information:

- Initial context factory supplied by the JNDI service provider
- Service provider URL for accessing the JNDI services
- Entity bean JNDI name
- Operation type (CREATE, UPDATE, or REMOVE)
- Finder method name (UPDATE and REMOVE operation types only)

Defining Entity Bean Jobs

You can define an Entity Bean (EJBE_JOB) 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 Statements

To define an Entity Bean job, you must specify the following statements:

- AGENT (see page 226)
- BEAN (see page 274)
- INITIAL_CONTEXT (see page 382)
- LOCATION (see page 413)
- OPERATIONTYPE (see page 435)

In addition, if the OPERATIONTYPE is UPDATE, you must also specify the following statements:

- FINDERMETHOD (see page 378)
- FINDERPARAMETER (see page 379)
- METHOD (see page 419)
- MODIFYPARAMETER (see page 423)

If the OPERATIONTYPE is REMOVE, you must also specify the following statements:

- FINDERMETHOD (see page 378)
- FINDERPARAMETER (see page 379)

Optional Statements

You can specify the following optional statements for an Entity Bean job:

- JNDIUSER (see page 387)
- JOBCLASS

In addition, if the OPERATIONTYPE is CREATE, you can also specify the following statements:

- CREATEMETHOD (see page 302)
- CREATEPARAMETER (see page 303)

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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
CREATEMETHOD createcustomer
OPERATIONTYPE CREATE
CREATEPARAMETER TYPE(String) VALUE(customerid)
CREATEPARAMETER TYPE(String) VALUE(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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE UPDATE
METHOD changephone
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
MODIFYPARAMETER TYPE(String) VALUE(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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE REMOVE
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
```

HTTP Jobs

The HTTP job invokes a program over HTTP in a similar way to a web browser. For example, you can use the HTTP job to invoke a CGI script, a Perl script, or a servlet. The HTTP job sends a URL over HTTP using the GET method or a form over HTTP using the POST method. The output of the invocation is returned in the job's spool file.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

The GET method requests data and sends the data as part of the URL. The POST method submits data and is the preferred method for sending lengthy form data.

To define an HTTP job, you require the following information:

- URL of the application server
- Program or servlet to invoke

Note: If your company has a firewall and you must communicate through a proxy server to access a computer outside the firewall, agent configuration is required. For more information on configuring the agent for a proxy, see the *CA Workload Automation Agent for Application Services Implementation Guide*.

Defining HTTP Jobs

You can define an HTTP (HTTP_JOB) 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 Statements

To define an HTTP job, you must specify the following statements:

- AGENT (see page 226)
- INVOCATIONTYPE (see page 385)
- SERVLET_URL (see page 579)

Optional Statements

You can specify the following optional statements for an HTTP job:

- ACTION (see page 225)
- AUTHORDER (see page 266)
- CONNECTIONDOMAIN (see page 293)
- CONNECTIONORIGIN (see page 294)
- CONNECTIONUSER (see page 295)
- FILTER (see page 375)
- JOBCLASS
- PARAMETER (see page 448)
- PROXYDOMAIN (see page 510)
- PROXYHOST (see page 511)
- PROXYORIGIN (see page 512)
- PROXYPORT (see page 514)
- PROXYUSER (see page 515)

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.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://google.com/search
PARAMETER KEYWORD(q) VALUE('ca workload automation')
AUTHORDER (BASIC, DIGEST, NTLM)
FILTER .*AE.*
```

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.

```
AGENT APPAGENT
INVOCATIONTYPE POST
SERVLET_URL http://localhost:8080
ACTION /examples/servlets/servlet/TheServlet
PARAMETER KEYWORD(key1) VALUE(subscribe)
PARAMETER KEYWORD(key2) VALUE(test@example.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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

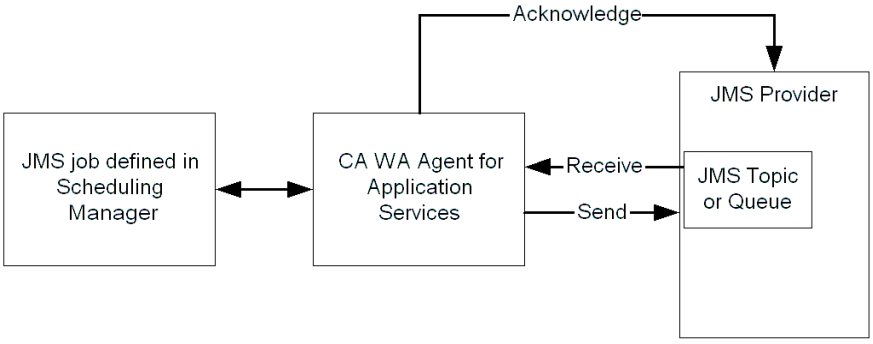
```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

JMS Publish and JMS Subscribe Jobs

Java Message Service (JMS) is the standard for enterprise messaging that lets a Java program or component (JMS client) produce and consume messages. Messages are the objects that communicate information between JMS clients.

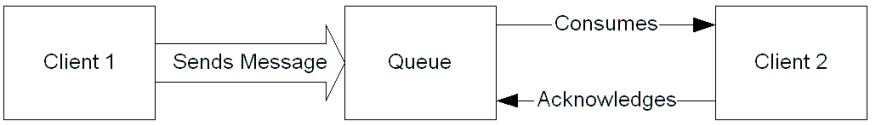
In a JMS system, a messaging server known as the JMS provider acts between two JMS clients (the publisher and the subscriber). Publishers send messages to the JMS provider while subscribers receive messages from the JMS provider.

The following diagram shows the functional relationship between the scheduling manager, the CA WA Agent for Application Services, and a JMS provider:



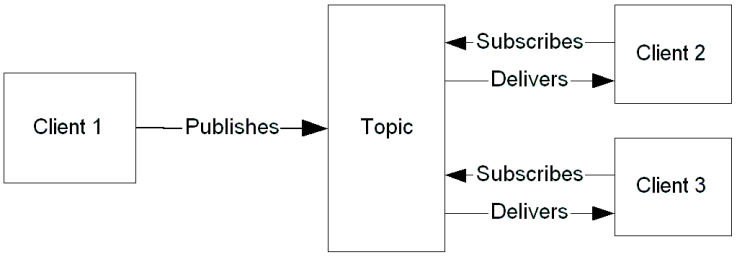
A queue is an object on the JMS server that holds messages sent by a client that are waiting to be consumed by another client. The queue retains a message until the message is consumed or the message expires.

The following diagram shows Client 2 (the subscriber) consuming a message that Client 1 (the publisher) sends to a queue:



A topic is an object a client uses to specify the target of the messages it produces and the source of the messages it consumes. A client acquires a reference to a topic on a JMS server, and sends messages to that topic. When messages arrive for that topic, the JMS provider is responsible for notifying all clients.

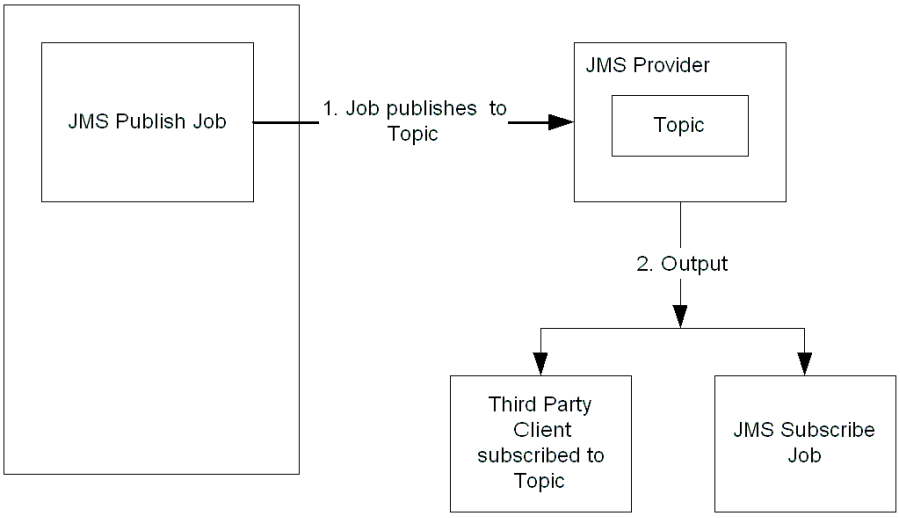
The following diagram shows two subscribers, Client 2 and Client 3, subscribed to a topic that the publisher, Client 1, publishes to:



A JMS Publish job lets you send a message to a queue or publish a message to a topic. Using a JMS Publish job to publish to a topic, you can broadcast a message to any topic subscriber. A third-party client can consume this message, or a JMS Subscribe job can listen for a particular message (using a filter).

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

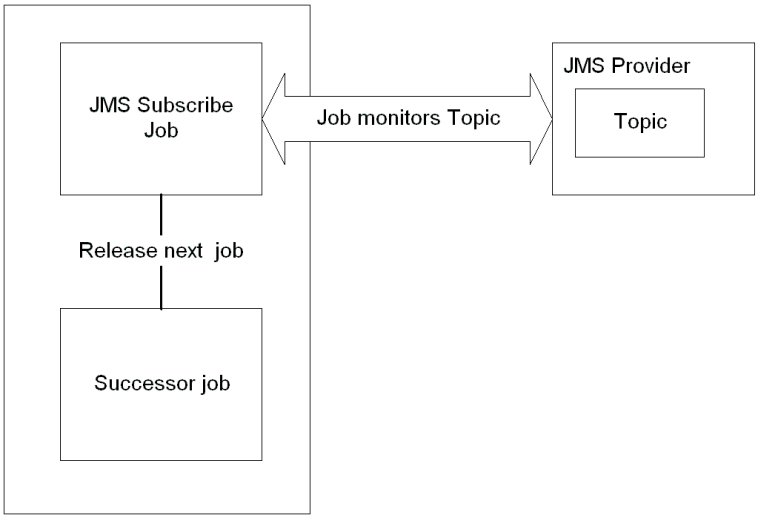
The following diagram shows a JMS Publish job scenario:



A JMS Subscribe job lets you consume messages from a queue or topic. Using a filter that you define within the job definition, the agent monitors the topic or queue output for specific data. The scheduling manager then sends the message that meets the filter criteria to a destination file you specify. You can define the job to continuously monitor JMS messages.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

The following diagram shows a JMS Subscribe job scenario:



To define a JMS Publish or JMS Subscribe job, you require the following information:

- Initial context factory supplied by the Java Naming and Directory Interface (JNDI) service provider
- JMS provider URL for accessing the JNDI services
- Connection factory JNDI name that looks up the referenced topic or queue
- JNDI name of the topic or queue on the JMS server
- Java class of the JMS message to send or publish

Defining JMS Publish Jobs

You can define a JMS Publish (JMSP_JOB) 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 Statements

To define a JMS Publish job, you must specify the following statements:

- AGENT (see page 226)
- CONNECTION_FACTORY (see page 292)
- DESTNAME (see page 316)
- INITIAL_CONTEXT (see page 382)
- LOCATION (see page 413)
- MSGCLASS (see page 429)
- TOPIC (see page 618)

Optional Statements

You can specify the following optional statements for a JMS Publish job:

- JNDIUSER (see page 387)
- JOBCLASS
- PARAMETER (see page 448)

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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

Defining JMS Subscribe Jobs

You can define a JMS Subscribe (JMSS_JOB) 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 Statements

To define a JMS Subscribe job, you must specify the following statements:

- AGENT (see page 226)
- CONNECTION_FACTORY (see page 292)
- DESTNAME (see page 316)
- INITIAL_CONTEXT (see page 382)
- LOCATION (see page 413)
- TOPIC (see page 618)

Optional Statements

You can specify the following optional statements for a JMS Subscribe job:

- DESTINATION (see page 314)
- FILTER (see page 375)
- JNDIUSER (see page 387)
- JOBCLASS

Example: Monitor a Queue on a WebLogic Application Server

This example continuously monitors the queue named Queue (residing on WebLogic) for messages 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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://172.24.0.0:7001
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue CONTINUOUS(a13)
FILTER abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+
TOPIC N
DESTINATION /export/home/user1/outputfile1
JNDIUSER 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

JMX Jobs

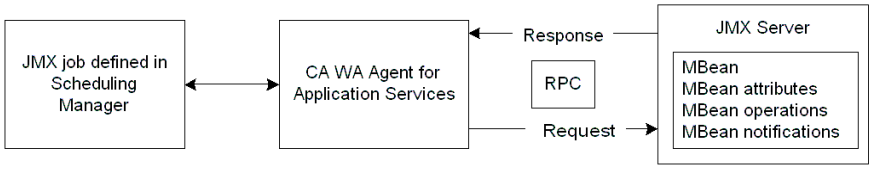
Java Management Extension (JMX) technology is included in the Java Standard Edition (SE) platform, version 5 and higher. JMX lets you remotely access applications, using a Remote Method Invocation (RMI) connector, for monitoring and management purposes.

JMX jobs let you access a remote JMX server that advertises MBeans. An MBean is a managed bean (Java object) that represents an application, a device, or any resource that you want to manage. An MBean contains a set of attributes and a set of operations that can be invoked. Some MBeans can send out notifications, for example, when an attribute changes.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Consider an MBean named Config that represents an application's configuration. The configuration parameters within that application are represented in Config by a set of attributes. Getting the attribute named cachesize, for example, returns the current value of the cachesize. Setting the value updates the cachesize. The Config MBean can send out a notification every time the cachesize changes. An operation named update, for example, can save changes to the configuration parameters.

The following diagram shows the functional relationship between the scheduling manager, CA WA Agent for Application Services, and the JMX server:



The JMX jobs provide support for getting and setting JMX MBean attributes, invoking JMX MBean operations, subscribing to MBean notifications, and creating and removing instances of MBeans on a JMX server.

You can define the following six types of JMX jobs:

- JMX-MBean Attribute Get
- JMX-MBean Attribute Set
- JMX-MBean Create Instance
- JMX-MBean Operation
- JMX-MBean Remove Instance
- JMX-MBean Subscribe

The JMX-MBean Attribute Set, JMX-MBean Create Instance, and JMX-MBean Operation jobs support calls to MBeans that can involve passing parameters. Each parameter can be an actual value or a serialized Java object passed by another job. When the JMX-MBean Operation job invokes an operation on an MBean that passes parameters, the parameters are passed to the MBean and the returned serialized Java object is stored on the agent computer in the spool directory or in a destination file you specify.

To define JMX jobs, you require a URL to connect to the JMX server using an RMI connector.

Defining JMX-MBean Attribute Get Jobs

You can define a JMX-MBean Attribute Get (JMXB_JOB) 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 Statements

To define a JMX-MBean Attribute Get job, you must specify the following statements:

- AGENT (see page 226)
- ATTRIBUTE (see page 264)
- MBEAN (see page 417)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Attribute Get job:

- DESTINATION (see page 314)
- FILTER (see page 375)
- JOBCLASS (see page 53)
- USER

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
ATTRIBUTE cachesize
```

Defining JMX-MBean Attribute Set Jobs

You can define a JMX-MBean Attribute Set (JMXA_JOB) 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 Statements

To define a JMX-MBean Attribute Set job, you must specify the following statements:

- AGENT (see page 226)
- ATTRIBUTE (see page 264)
- MBEAN (see page 417)
- PARAMETER (see page 448)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Attribute Set job:

- DESTINATION (see page 314)
- FILTER (see page 375)
- JOBCLASS
- USER

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 located in a path on the agent computer.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
MBEAN 'DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic'
ATTRIBUTE State
PARAMETER URI('FS:C:\Program Files\CA\WA Agent +
R11.3_B129\spool\CM_QABC\MAIN\LLR113.7\JMXATT.GET.12587292392520')
```

Defining JMX-MBean Create Instance Jobs

You can define a JMX-MBean Create Instance job (JMXN_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 Statements

To define a JMX-MBean Create Instance job, you must specify the following statements:

- AGENT (see page 226)
- CLASSNAME (see page 281)
- MBEAN (see page 417)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Create Instance job:

- JOBCLASS
- PARAMETER (see page 448)
- USER

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".

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
MBEAN 'DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic'
CLASSNAME cdc.jmx.SimpleDynamic
PARAMETER TYPE(java.lang.String) VALUE(Hello)
```

Defining JMX-MBean Operation Jobs

You can define a JMX-MBean Operation (JMXO_JOB) 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 Statements

To define a JMX-MBean Operation job, you must specify the following statements:

- AGENT (see page 226)
- MBEAN (see page 417)
- OPERATION (see page 434)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Operation job:

- DESTINATION (see page 314)
- FILTER (see page 375)
- JOBCLASS
- PARAMETER (see page 448)
- USER

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
OPERATION resetmem
PARAMETER TYPE(Integer) VALUE(50)
```

Defining JMX-MBean Remove Instance Jobs

You can define a JMX-MBean Remove Instance (JMXR_JOB) 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 Statements

To define a JMX-MBean Remove Instance job, you must specify the following statements:

- AGENT (see page 226)
- MBEAN (see page 417)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Remove Instance job:

- JOBCLASS
- USER

Example: Remove an MBean Instance from a JMX Server

Suppose that you want to remove an MBean instance.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
MBEAN 'DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic'
```

Defining JMX-MBean Subscribe Jobs

You can define a JMX-MBean Subscribe (JMXS_JOB) 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 Statements

To define a JMX-MBean Subscribe job, you must define the following statements:

- AGENT (see page 226)
- MBEAN (see page 417)
- URL (see page 633)

Optional Statements

You can specify the following optional statements for a JMX-MBean Subscribe job:

- ATTRIBUTESFILTER (see page 265)
- JOBCLASS
- TYPESFILTER (see page 630)
- USER

Example: Monitor for Changes to a Specific MBean Attribute

Suppose that you want to set up continuous monitoring for changes to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. Each time the cachesize attribute changes, an alert named CHGA is sent.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config' CONTINUOUS(CHGA)
ATTRIBUTESFILTER 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 named CHGA is sent.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config' CONTINUOUS(CHGA)
TYPESFILTER jmx.attribute.change
```

POJO Jobs

A Plain Old Java Object (POJO) is a Java object that follows the Java Language Specification only. All Java objects are POJOs.

The POJO job lets you instantiate a class to create a Java object and invoke a method on it. The job is restricted to classes that take constructors with no arguments (default constructors).

Note: To run these jobs, your system requires the following agents:

- CA WA Agent for UNIX, Linux, or Windows
- CA WA Agent for Application Services or CA WA Agent for Web Services

You can use the POJO job to invoke custom Java code on a local computer. POJO jobs support method calls that can involve passing parameters. The parameters can be actual values or a serialized Java object passed by another job. When the POJO job invokes a method on an object, the parameters, if any, are passed to the object and the returned values are stored in a Java serialized object file.

To define a POJO job, you require the class name and method you want to call on the instantiated object.

Notes:

- If you use custom Java code, verify that the required JAR file is in the jars subdirectory of the agent installation directory.
- By default, the oscomponent.classpath and oscomponent.javapath environment variables are set in the agentparm.txt file when you install the agent. You cannot modify environment variables using a POJO job. But you can run a batch file using a Windows job to set the CLASSPATH and JAVA_HOME environment variables, and then run the Java code.
- After you change or redeploy a class or JAR file in the agent installation directory, restart the agent. For more information about restarting the agent, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

- The JRE version of the agent that you use to run POJO jobs and the Java compiler on your computer must be the same. To find the JRE version of the agent, follow these steps:

  1. Change to the following directory at the command prompt:

     - On Windows:

     agent_install_dir\jre\bin

     - On UNIX:

     agent_install_dir/jre/bin

     agent_install_dir

     Specifies the agent installation directory.

  2. Enter the following command:

     java.exe –version

Defining POJO Jobs

You can define a POJO (POJO_JOB) 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 the following agents:

- CA WA Agent for UNIX, Linux, or Windows
- CA WA Agent for Application Services or CA WA Agent for Web Services

Required Statements

To define a POJO job, you must specify the following statements:

- AGENT (see page 226)
- CLASSNAME (see page 281)
- METHOD (see page 419)

Optional Statements

You can specify the following optional statements for a POJO job:

- DESTINATION (see page 314)
- JOBCLASS
- PARAMETER (see page 448)
- USER

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.

```
AGENT APPAGENT
CLASSNAME java.lang.Integer
METHOD parseInt
PARAMETER TYPE(java.lang.String) VALUE(5)
```

RMI Jobs

Remote Method Invocation (RMI) is the Java version of a Remote Procedure Call (RPC), which is a technology that lets a program request a service from another program located in another address space. That address space could be on the same computer or on a different one.

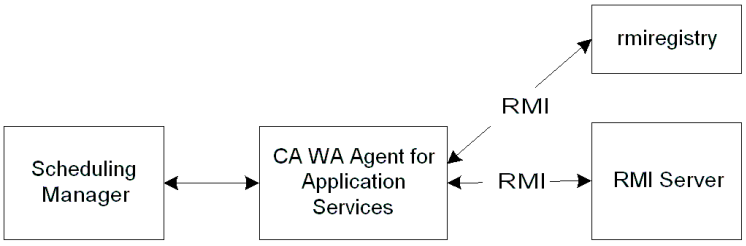
RMI jobs let you set up interaction between Java objects on different computers in a distributed network. Using an RMI job, you can access a remote server and invoke a method on a Java object. A method is a programmed procedure that is defined as a part of a Java class.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

RMI jobs support method calls to remote objects that can involve passing parameters. The parameters can be actual values or a serialized Java object passed by another job. When the RMI job invokes a method on an object that passes parameters, the parameters are passed to the remote object and the returned serialized Java object is stored on the agent computer in the spool directory or in a destination file you specify.

RMI uses a naming or directory service to locate the remote object on the remote server. To define an RMI job, you require the naming class of the Java object you want to invoke a method on. That naming class takes a name that is a java.lang.String in URL format.

The following diagram shows the functional relationship between the scheduling manager, CA WA Agent for Application Services, and an RMI Server:



Defining RMI Jobs

You can define an RMI (RMI_JOB) 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 Statements

To define an RMI job, you must specify the following statements:

- AGENT (see page 226)
- METHOD (see page 419)
- REMOTECLASSNAME (see page 528)

Optional Statements

You can specify the following optional statements for an RMI job:

- DESTINATION (see page 314)
- JOBCLASS
- PARAMETER (see page 448)

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.

```
AGENT APPAGENT
REMOTECLASSNAME rmi://remotehost/Test
METHOD startserver
PARAMETER TYPE(String) VALUE(now)
```

Session Bean Jobs

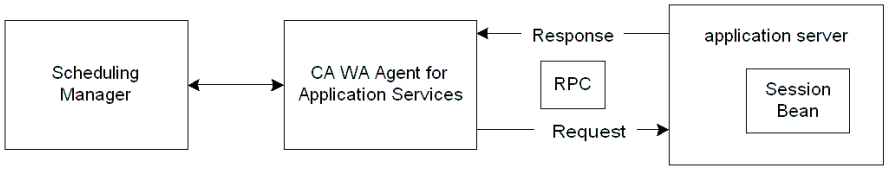
A session bean represents business logic or action to be taken (for example, charging a credit card or adding items to an online shopping cart).

Unlike entity beans, which are stored in a database, session beans may be destroyed after each use. For example, when a session bean is invoked to perform credit card validation, the application server creates an instance of that session bean, performs the business logic to validate the credit card transaction, and then destroys the session bean instance after the credit card transaction has been validated.

You can use a session bean under the following conditions:

- The bean represents a procedure and not a business entity. For example, you use a session bean to encrypt data or add items to an online shopping cart.
- The state of the bean does not have to be kept in permanent storage. For example, when the bean instance terminates or the application server shuts down, the bean's state is no longer required.

The following diagram shows the functional relationship between the scheduling manager, CA WA Agent for Application Services, and a session bean residing on an application server:



The Session Bean job lets you access a session bean on an application server. This job type can make a Remote Procedure Call (RPC) to the session bean, invoke a method that defines the business logic, pass parameters to the method, and have the results returned as serialized Java output. The output can be stored on the agent computer as text in the spool file or as a serialized Java object in the spool directory or a destination file you specify.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

You can access stateless and stateful session beans using the Session Bean job. The job acts in a similar way for both types of beans. For both stateful and stateless beans, you can specify parameters to pass to the method. When you define a stateful session bean, however, you must specify parameters to define the bean. After the method is invoked, the agent destroys the stateful bean.

Use a stateless Session Bean job to invoke a single instance of a method on the bean, such as encrypting data or sending an email to confirm an order. Use a stateful Session Bean job to invoke the same method on the bean multiple times, such as adding multiple items to an online shopping cart.

A Session Bean job requires a dedicated connection between the agent and the application server. To define a Session Bean job, you require the following information:

- Initial context factory supplied by the Java Naming and Directory Interface (JNDI) service provider
- Service provider URL for accessing the JNDI services
- Session bean JNDI name
- Method to be invoked

Defining Session Bean Jobs

You can define a Session Bean (EJB_JOB) 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 Statements

To define a Session Bean job, you must specify the following statements:

- AGENT (see page 226)
- BEAN (see page 274)
- INITIAL_CONTEXT (see page 382)
- LOCATION (see page 413)
- METHOD (see page 419)

In addition, to access stateful session beans, you require the following statements:

- CREATEMETHOD (see page 302)
- CREATEPARAMETER (see page 303)

Optional Statements

You can specify the following optional statements for a Session Bean job:

- DESTINATION (see page 314)
- JNDIUSER (see page 387)
- JOBCLASS
- PARAMETER (see page 448)

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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN Shoppingcart
CREATEMETHOD createaddbook
METHOD addbook
CREATEPARAMETER TYPE(String) VALUE(ISBN)
PARAMETER TYPE(Integer) ARRAY(1551929120,1582701709)
```

Chapter 3: Database Jobs

This section contains the following topics:

Database Jobs (see page 67)
How Database Trigger Jobs Differ from Database Monitor Jobs (see page 68)
Defining Database Monitor Jobs (see page 69)
Defining Database Stored Procedure Jobs (see page 71)
Defining Database Trigger Jobs (see page 76)
Defining SQL Jobs (see page 81)

Database Jobs

Database jobs let you automate common database tasks on Oracle, Microsoft SQL Server, and IBM DB2 databases.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

You can define the following database jobs:

Database Monitor

Lets you monitor for an increase or decrease in the number of rows in a database table.

Database Stored Procedure

Lets you run a stored procedure.

Database Trigger

Lets you monitor for added, deleted, and updated rows in a database table.

SQL

Lets you execute an SQL statement.

How Database Trigger Jobs Differ from Database Monitor Jobs

You can monitor database tables using Database Monitor or Database Trigger jobs.

A Database Monitor job can monitor a database table for an increase or decrease in the number of rows. To monitor the database for specific changes, you can add a monitor condition to the job definition. Database Monitor jobs count the number of rows that satisfy the monitor condition using a polling interval, which is every 10 seconds by default. Database Monitor jobs do not detect other updates to a row or changes to the number of rows that cancel each other out during the polling interval. For example, suppose that within a 10-second interval, a row is added while another row is deleted. Since the total number of rows did not change, the Database Monitor job does not detect the row addition and deletion.

Alternatively, a Database Trigger job can monitor a database table for added rows, deleted rows, or updated rows. To monitor the database for specific changes, you can add a trigger condition to the job definition. Each Database Trigger job creates a database trigger on the database. The database trigger templates that the agent uses are located in the directory where the agent is installed. The template files are named dbtrig*database_type*.properties, for example, dbtrigOracle.properties. Contact your database administrator before choosing to use a Database Trigger job.

For either job type, you can set up continuous monitoring so that each time a database change occurs, an alert or an event is triggered.

Note: Not all scheduling managers support alerts and events.

Important! You should not drop a table that is being referenced by either a Database Monitor or a Database Trigger job. The job will remain active, even though the table has been dropped.

Defining Database Monitor Jobs

You can define a Database Monitor (DB_MON) 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. You can set up continuous monitoring so that each time a database change occurs, an alert is triggered. For continuous monitoring, the job state changes to a monitoring state and remains in that state until it is forced complete or cancelled.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Statements

To define a Database Monitor job, you must specify the following statements:

- AGENT (see page 226)
- MON_TYPE (see page 426)
- TABLE_NAME (see page 599)

Note: All database jobs require a database user ID and the database resource location. You can use the defaults defined on the agent in the agentparm.txt file or specify the USER and DB_URL statements in the job definition.

Optional Statements

You can specify the following optional statements for a Database Monitor job:

- DB_URL (see page 308)
- JOBCLASS
- MON_COND (see page 425)
- USER
- USER_TYPE (see page 638)

Example: Monitor a Table for a Change in the Number of Rows

Suppose that you want a job to monitor the STAFF table for a change in the number of rows. When a row that has the name Jonson is added or deleted, the job completes. The job uses the database resource location default defined on the agent.

```
AGENT DB_AGENT
TABLE_NAME STAFF
MON_TYPE ID
MON_COND NAME='Jonson'
USER entadm
```

Example: Monitor a Table for Increases in the Number of Rows

Suppose that you want a job to monitor the emp table for increases in the number of rows. When a new row has a salary greater than 100000, the scheduling manager sends an alert. The job remains in a monitoring state until it is forced complete or cancelled. The job uses the user name and database resource location defaults defined on the agent.

```
AGENT DB_AGENT
MON_TYPE I
TABLE_NAME emp CONTINUOUS(high)
MON_COND sal>100000
```

Defining Database Stored Procedure Jobs

You can define a Database Stored Procedure (DBSP_JOB) 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.

If you use Oracle or SQL Server, 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 Statements

To define a Database Stored Procedure job, you must specify the following statements:

- AGENT (see page 226)
- STORED_PROCEDURE (see page 597)

Note: All database jobs require a database user ID and the database resource location. You can use the defaults defined on the agent in the agentparm.txt file or specify the USER and DB_URL statements in the job definition.

Optional Statements

You can specify the following optional statements for a Database Stored Procedure job:

- ARGS (see page 253)
- DB_URL (see page 308)
- JOB_CRITERIA (see page 389)
- JOBCLASS
- RETURN_DATA_TYPE (see page 540)
- USER
- USER_TYPE (see page 638)

Example: Invoke a Simple Stored Procedure from a Microsoft SQL Server Database

Suppose that you want a job to run the byroyalty stored procedure located in the pubs sample database. When the job runs, the agent passes the input parameter percentage a value of 40.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SPNAME byroyalty
ARGS percentage IN INTEGER,40
```

Example: Invoke a Stored Procedure with Input and Output Parameters from a Microsoft SQL Server Database

Suppose that you want a job to run the emp stored procedure, which returns a value from the emp table. The pubid returned matches the employee named Ann Devon. The pubid, 9952, appears in the job's spool file.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SPNAME emp
ARGS f_name IN VARCHAR,Ann
ARGS l_name IN VARCHAR,Devon
ARGS pubid OUT CHAR
```

To run the job in this example, create the emp stored procedure in the pubs database using the following script:

```
CREATE PROCEDURE EMP
(@f_name VARCHAR(20),
@l_name VARCHAR(30),
@pubid CHAR(4) OUTPUT)
AS BEGIN
SELECT
@pubid=pub_id
FROM emp
WHERE
fname=@f_name
and
lname=@l_name
print @l_name+@f_name+@pubid
END
GO
```

Example: Invoke a Stored Procedure with Input and Output Parameters from an IBM DB2 Database

Suppose that you want a job to run the DEPT_MEDIAN stored procedure under the user entadm. DEPT_MEDIAN returns the median salary of department 20 from the STAFF table. The median salary, 18171.25, appears in the job's spool file.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
STORED_PROCEDURE ENTADM.DEPT_MEDIAN
ARGS deptNumber IN SMALLINT,20
ARGS medianSalary OUT DOUBLE
```

The spool file for this job contains the following output:

```
{ call ENTADM.DEPT_MEDIAN(?, ?) }
medianSalary=18171.25
```

The job in this example runs the following stored procedure in the SAMPLE database:

```
CREATE PROCEDURE DEPT_MEDIAN
  (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
  LANGUAGE SQL
  BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
      SELECT CAST(salary AS DOUBLE) FROM staff
        WHERE DEPT = deptNumber
        ORDER BY salary;
      DECLARE EXIT HANDLER FOR NOT FOUND
        SET medianSalary = 6666;
-- initialize OUT parameter
        SET medianSalary = 0;
        SELECT COUNT(*) INTO v_numRecords FROM staff
          WHERE DEPT = deptNumber;
        OPEN c1;
        WHILE v_counter < (v_numRecords / 2 + 1) DO
          FETCH c1 INTO medianSalary;
          SET v_counter = v_counter + 1;
        END WHILE;
        CLOSE c1;
END
```

Supported Data Types

The following table lists the supported data types of 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	MS SQL Server	DB2
CHAR	Plain text	CHAR	CHAR	CHAR
VARCHAR	Plain text	VARCHAR2(x) VARCHAR(x)	VARCHAR(x)	VARCHAR(x)
LONGVARCHAR		Not supported	Not supported	Not supported
NUMERIC	Number	NUMBER, 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 INT	INTEGER
BIGINT	Number	Not supported	BIGINT	BIGINT
REAL	Number [dot Number]	REAL	REAL	REAL
FLOAT	Number [dot Number]	FLOAT	FLOAT	FLOAT
DOUBLE			FLOAT	DOUBLE
DATE	yyyy-mm-dd	DATE	Not supported	DATE
	String (e.g., SYSDATE)			
TIME	hh:mm:ss		Not supported	TIME
TIMESTAMP	yyyy-mm-dd hh:mm:ss.fffffff	TIMESTAMP	datetime smalldatetime	TIMESTAMP

JDBC Data Type	Valid Input Format	Oracle	MS SQL Server	DB2
BINARY	Hex (e.g., 010203FF)		timestamp	

Note: 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 input values to values supported by the database driver and calls the appropriate stored procedure from inside the database.

Example: Run a Wrapper Procedure

In the following example, the agent runs a wrapper procedure called 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;
```

Defining Database Trigger Jobs

You can define a Database Trigger (DB_TRIG) 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. You can set up continuous monitoring so that each time a database change occurs, an alert is triggered. For continuous monitoring, the job state changes to a monitoring state and remains in that state until it is forced complete or cancelled.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Required Statements

To define a Database Trigger job, you must specify the following statements:

- AGENT (see page 226)
- TABLE_NAME (see page 599)
- TRIG_TYPE (see page 628)

Note: All database jobs require a database user ID and the database resource location. You can use the defaults defined on the agent in the agentparm.txt file or specify the USER and DB_URL statements in the job definition.

Optional Statements

You can specify the following optional statements for a Database Trigger job:

- DB_URL (see page 308)
- JOBCLASS
- TRIG_COND (see page 626)
- USER
- USER_TYPE (see page 638)

Examples: Monitoring Oracle Database Tables

This topic contains examples of Database Trigger jobs that monitor Oracle database tables.

Example: Monitor an Oracle Database Table for an Added or Deleted Row

Suppose that you want a job to monitor the emp table for an added row or a deleted row. The job runs under the user named scott, who has the authority to create triggers on the schema the table belongs to. The job remains in a running state while waiting for an added or deleted row. When a row is either added or deleted, the job completes.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
TRIG_TYPE 'INSERT OR DELETE'
TABLE_NAME emp
```

Example: Monitor an Oracle Database Table for Deleted Rows with a Trigger Condition

Suppose that you want a job to monitor the emp table for deleted rows. The job runs under a user who has the authority to create triggers on the schema the table belongs to. When a row containing department 75 is deleted, the job completes.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
TRIG_TYPE DELETE
TABLE_NAME emp
TRIG_COND old.deptno=75
```

Example: Monitor an Oracle Database Table for Added Rows with a Trigger Condition

Suppose that you want a job to monitor the emp table for added rows. The job runs under a user who has the authority to create triggers on the schema the table belongs to. When a row containing a name beginning with the letter g is added, the job completes.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
TRIG_TYPE INSERT
TABLE_NAME emp
TRIG_COND 'new.ename like ''g%%'''
```

Example: Monitor an Oracle Database Table for Added or Updated Rows with a Trigger Condition

Suppose that you want a job to monitor the emp table for added or updated rows. The job runs under a user who has the authority to create triggers on the schema the table belongs to. The job completes when a new or updated row does not contain a job field equal to sales.

Note: The <> symbol indicates not equal to.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
TRIG_TYPE 'INSERT OR UPDATE'
TABLE_NAME emp
TRIG_COND new.job<>'sales'
```

Example: Monitor an Oracle Database Table for Deleted Rows

Suppose that you want a job to monitor the emp table for deleted rows. When a row is deleted, the job changes from a running state to a monitoring state. Each time a row is deleted from the table, the scheduling manager sends the alert named alrt. The job remains in a monitoring state until it is forced complete or cancelled. The job runs under the user named scott, who has the authority to create triggers on the schema the table belongs to.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
TRIG_TYPE DELETE
TABLE_NAME emp CONTINUOUS(alrt)
```

Examples: Monitoring Microsoft SQL Server Database Tables

This topic contains examples of Database Trigger jobs that monitor Microsoft SQL Server database tables.

Example: Monitor a Microsoft SQL Server Database Table for a New or Deleted Row

Suppose that you want a job to monitor the stores table for an added row or a deleted row. The job runs on the DB_AGENT agent computer under the sa user. 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.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
TRIG_TYPE 'INSERT, DELETE'
TABLE_NAME stores
```

Example: Monitor a Microsoft SQL Server Database Table for Two Changes

Suppose that you want a job to monitor the sales table for changes to the ord_date and qty columns. The job runs under the sa user and completes only when both columns have changed.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
TRIG_TYPE UPDATE
TABLE_NAME SALES
TRIG_COND 'UPDATE(ord_date) and UPDATE(qty)'
```

Example: Monitor a Microsoft SQL Server Database Table for Added Rows with a Trigger Condition

Suppose that you want a job to monitor the sales table for added rows. When the quantity for inserted title ID TC7777 is greater than or equal to 20, the job completes.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
TRIG_TYPE INSERT
TABLE_NAME sales
TRIG_COND '(select QTY from INSERTED where TITLE_ID=''TC7777'')>=20'
```

Example: Monitor a Microsoft SQL Server Database Table for Deleted Rows

Suppose that you want a job to monitor the sales table for deleted rows. the job changes from a running state to a monitoring state. Each time a row is deleted from the table, the scheduling manager sends the alert named altr. The job remains in a monitoring state until it is forced complete or cancelled.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
TRIG_TYPE DELETE
TABLE_NAME sales CONTINUOUS(altr)
```

Examples: Monitoring IBM DB2 Database Tables

This topic contains examples of Database Trigger jobs that monitor IBM DB2 database tables.

Example: Monitor an IBM DB2 Database Table for Added Rows with a Trigger Condition

Suppose that you want a job to monitor the STAFF table for added rows. When the total number of rows is greater than or equal to 37, the job completes.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
TRIG_TYPE INSERT
TRIG_COND '(select count(*) from STAFF)>=37'
TABLE_NAME STAFF
```

Example: Monitor an IBM DB2 Database Table for Changes

Suppose that you want to monitor the STAFF table for changes to the DEPT and JOB columns. The job runs on the DB_AGENT agent computer and connects to the SAMPLE database. The job runs under the entadm user and completes once DEPT or JOB is updated.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
TABLE_NAME STAFF
TRIG_TYPE 'UPDATE of DEPT, JOB'
```

Defining SQL Jobs

You can define an SQL (SQL_JOB) job to run an SQL query against an Oracle, Microsoft SQL Server, or IBM 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. 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 Statements

To define an SQL job, you must specify the following statements:

- AGENT (see page 226)
- SQL (see page 591)

Note: All database jobs require a database user ID and the database resource location. You can use the defaults defined on the agent in the agentparm.txt file or specify the USER and DB_URL statements in the job definition.

Optional Statements

You can specify the following optional statements for an SQL job:

- DB_URL (see page 308)
- JOB_CRITERIA (see page 389)
- JOBCLASS
- OUTPUT_FILE (see page 443)
- USER
- USER_TYPE (see page 638)

Examples: Running SQL Statements Against Oracle Database Tables

This topic contains examples of SQL jobs that run SQL statements against Oracle database tables.

Example: Add a Row to an Oracle Database Table

Suppose that you want a job to add a row of data to the emp table.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
SQL 'INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, +
  COMM, DEPTNO) VALUES(2476, ''robert'', ''sales'', 435, +
  ''01-OCT-2005'', 65000, 10, 75)'
```

Example: Update a Row to an Oracle Database Table

Suppose that you want a job to update a record in the emp table and change the salary to 75,000 for the employee with name robert.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
SQL 'UPDATE EMP SET SAL=75000 where ENAME=''robert'''
```

Example: Delete a Row from an Oracle Database Table

Suppose that you want a job to delete a row from the emp table for the employee with name robert.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
SQL 'DELETE FROM EMP WHERE ENAME=''robert'''
```

Example: Return Data from an Oracle Database Table that Match a Condition

Suppose that you want a job to query the emp table for names that have salaries greater than 40,000. If the query returns a name that begins with the letter d, the job completes.

```
AGENT DB_AGENT
USER scott
DB_URL jdbc:oracle:thin:@myhost:1521:orcl
SQL 'SELECT ENAME FROM EMP WHERE SAL > 40000'
JOB_CRITERIA ENAME=d.*
OUTPUT_FILE /emp/salary.txt
```

For example, the salary.txt file contains the following output:

```
Output for: SELECT ENAME FROM EMP WHERE SAL > 40000

ENAME
-----------
donald
```

Examples: Running SQL Statements Against Microsoft SQL Server Database Tables

This topic contains examples of SQL jobs that run SQL statements against Microsoft SQL Server database tables.

Example: Add a Row to a Microsoft SQL Server Database Table

Suppose that you want a job to add a row for a new store to the stores table.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SQL 'INSERT INTO stores(stor_id, stor_name, stor_address, +
  city, state, zip) VALUES(''6523'', ''Chapters'', +
  ''6523 Main St.'', ''San Diego'', ''CA'', ''93223'')'
```

Example: Delete a Row from a Microsoft SQL Server Database Table

Suppose that you want a job to delete the row for store ID 6523 from the stores table.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SQL 'DELETE FROM stores WHERE stor_id=''6523'''
```

Example: Update a Row in a Microsoft SQL Server Database Table

Suppose that you want a job to update the row in the sales table that matches order number 6871 and change the values for the order date and quantity.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SQL 'UPDATE sales SET ord_date=''6/15/2006'', qty=10 WHERE +
  ord_num=''6871'''
```

Example: Return Data from a Microsoft SQL Server Database Table that Matches a Condition

Suppose that you want a job to query the sales table for orders that have a quantity greater than 20. The order numbers that match the query appear in the output file ordnum.txt.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
SQL 'SELECT ORD_NUM FROM SALES WHERE QTY > 20'
JOB_CRITERIA ORD_NUM=A2976
OUTPUT_FILE C:\sales\ordnum.txt
```

The ordnum.txt file contains the following order numbers:

```
A2976
QA7442.3
P2121
N914014
P3087a
P3087a
X999
P723
QA879.1
```

Because the query returns an order number that matches the job criteria A2976, the job completes.

Suppose we change the job criteria statement in the above example to the following:

```
JOB_CRITERIA .*B[0-9]
```

In this case, the query would still return the same order numbers, but the job would fail because it would not find a matching order number containing the letter B and followed by a number.

Examples: Running SQL Statements Against IBM DB2 Database Tables

This topic contains examples of SQL jobs that run SQL statements against IBM DB2 database tables.

Example: Add a Row to an IBM DB2 Database Table

Suppose that you want a job to add a row of data to the STAFF table under the user entadm.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
SQL 'INSERT into ENTADM.STAFF(ID, NAME, DEPT, JOB, YEARS, +
  SALARY, COMM) VALUES(556, ''Jonson'', 84, ''Sales'', 1, +
  40500.50, 100)'
```

Example: Update a Row in an IBM DB2 Database Table

Suppose that you want a job to update a record in the STAFF table under the user entadm. The job changes the years to 3 for the employee with the name Jonson.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
SQL 'UPDATE ENTADM.STAFF SET YEARS=3 where NAME=''Jonson'''
```

Example: Delete a Row from an IBM DB2 Database Table

Suppose that you want a job to delete a row from the STAFF table under the user entadm for the employee with the name Jonson.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
SQL 'DELETE FROM ENTADM.STAFF where NAME=''Jonson'''
```

Example: Return Data from an IBM DB2 Database Table that Matches a Condition

Suppose that you want a job to query the STAFF table under the user entadm for employees that have salaries greater than 40,000. If the query returns an employee name that begins with the letter J, the job completes.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
SQL 'SELECT NAME FROM ENTADM.STAFF where SALARY > 40000'
JOB_CRITERIA NAME=J.*
OUTPUT_FILE /staff/salary.txt
```

For example, the salary.txt file contains the following output:

```
Output for: SELECT NAME FROM ENTADM.STAFF where SALARY > 40000

NAME
-----------
Jonson
```

Chapter 4: File Trigger Jobs

This section contains the following topics:

File Trigger Jobs (see page 87)
Defining File Trigger Jobs (see page 88)
Monitor a File that is Owned by a UNIX Owner or Group (see page 90)
Monitor a File on a Remote Windows Computer (see page 91)

File Trigger Jobs

File Trigger jobs let you monitor file activity. You can define File Trigger jobs for UNIX, Linux, Windows, HP Integrity NonStop, or i5/OS systems.

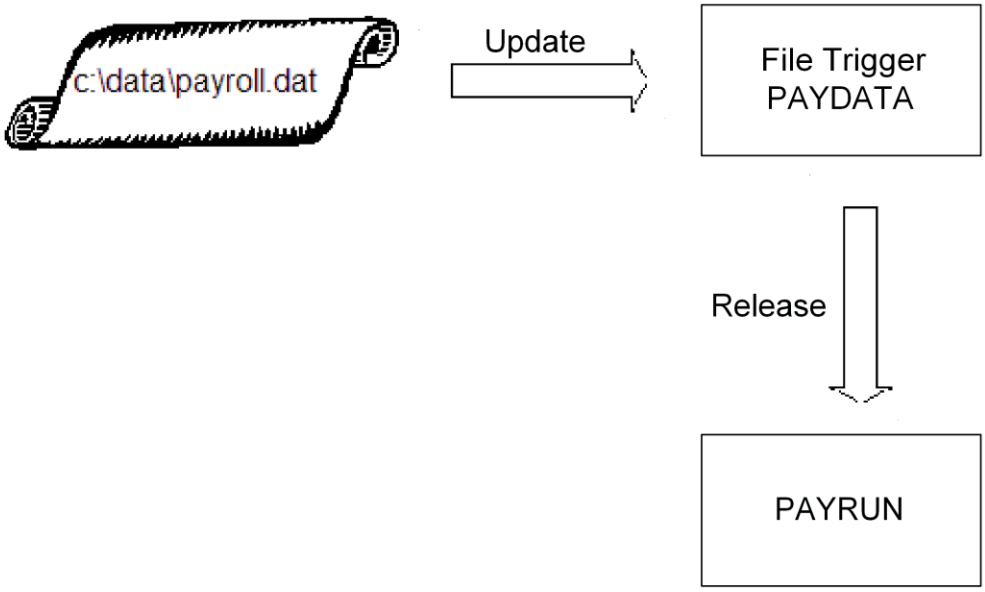
Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, HP Integrity NonStop, or i5/OS.

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.

Example: Monitor for an Update to a File

Suppose that a File Trigger job named PAYDATA monitors for an update to the payroll.dat file on a Windows computer. When the file is updated, the job completes and the scheduling manager releases a job named PAYRUN.

The following diagram shows the scenario:



Defining File Trigger Jobs

You can define a File Trigger (FILE_TRIGGER) job to 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, i5/OS, or HP Integrity NonStop.
- CA WA Agent for HP Integrity NonStop only supports monitoring of file creation or update.

Required Statements

To define a File Trigger job, you must specify the following statements:

- AGENT (see page 226)
- FILENAME (see page 358)

Optional Statements

You can specify the following optional statements for a File Trigger job:

- JOBCLASS
- USER

Example: Monitor an Update to a Payroll File on a Windows Computer

This example monitors an update to the payroll.dat file on a Windows computer. When the file is updated, the job completes.

```
AGENT NT_NY
FILENAME c:\data\payroll.dat UPDATE
```

Example: Monitor an Update to a Payroll File on a UNIX Computer

This example monitors an update to the payroll file on a UNIX computer. When the file is updated, the job completes.

```
AGENT UNIX_NY
FILENAME /usr/data/payroll UPDATE
```

Example: Monitor an Update to a Payroll File Object on an i5/OS System

This example monitors an update to the payroll file object on an i5/OS computer. When the file is updated, the job completes.

```
AGENT I5_NY
FILENAME /QSYS.LIB/LIBRARY.LIB/DEPT.FILE/PAYROLL.MBR UPDATE
```

Example: Monitor an Update to a Payroll File Object on an HP Integrity NonStop System

This example monitors an update to the payroll file object on an HP Integrity NonStop computer. When the file is updated, the job completes.

```
AGENT HPNS_NY
FILENAME $SYS.DATA.PAYROLL UPDATE
```

Monitor a File that is Owned by a UNIX Owner or Group

You can define a File Trigger job to monitor a file that is owned by a specific UNIX owner or group. If the file is not owned by the specified owner or group, the following occurs:

- The job continues monitoring if the file trigger type is CREATE.
- The job completes if the file trigger type is DELETE or NOTEXIST.
- The job fails if the file trigger type is EXPAND, EXIST, SHRINK, or UPDATE.

Note: You can monitor a file owned by a specific owner or group on i5/OS if the file is not a QSYS object.

To monitor a file that is owned by a UNIX owner or group, specify the OWNER or GROUP operands in the FILENAME statement in the job definition.

Example: Monitor for the Creation of a File that Is Owned by a Specified UNIX User ID

This example monitors for the creation of a file named payroll.dat owned by JDOE:

```
AGENT SYSAGENT
FILENAME /data/payroll.dat CREATE OWNER(JDOE)
```

The job completes as follows:

- If the file does not exist when the job is readied, 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.

Example: Monitor for the Existence of a File that Is Owned by a Specified UNIX Group

This example monitors for the existence of the payroll.dat file that is owned by the UNIX group ACCTS:

```
AGENT SYSAGENT
FILENAME /data/payroll.dat EXIST GROUP(ACCTS)
```

The job completes as follows:

- 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.

Monitor a File on a Remote Windows Computer

You can define a File Trigger job to monitor a file on a remote Windows system if the agent runs as a Windows service under the local system account.

To monitor a file on a remote Windows computer

1. Verify with your agent administrator that the agent is running as a Windows service under the local system account.
2. Ask your scheduling manager administrator to define a user ID and password on the scheduling manager that has access to the file on the remote Windows computer.
3. Specify the file name to monitor in the FILENAME statement in the job definition using UNC (Universal Naming Convention). 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.
4. Specify the user ID and the domain the user ID belongs to in the USER statement.

Example: Monitor for an Update to a File on a Remote Windows System

This example monitors for an update to the payroll.dat file on a remote Windows computer named CYBNT. The job runs under JSMITH, which is a user ID on CYBNT and is in the CYBDOM domain. JSMITH is defined on the scheduling manager and has access to the AccountingFiles directory.

```
AGENT SYSAGENT
FILENAME \\CYBNT\AccountingFiles\payroll.dat UPDATE
USER CYBDOM\JSMITH
```

Chapter 5: FTP Jobs

This section contains the following topics:

FTP Jobs (see page 93)
Defining FTP Jobs (see page 96)
Transfer Files Using SSL FTP (see page 99)
Compress Data for FTP (see page 101)
Send Site-Specific FTP Commands to FTP Servers (see page 102)
Verify the FTP Job Status (see page 103)

FTP Jobs

Using your agent, you can automate File Transfer Protocol (FTP) transfers with an FTP job. The FTP job can upload data to or download data from an existing FTP server or another agent running 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.

You can use an FTP job to automate the following:

- Download ASCII, binary, or EBCDIC (i5/OS only) files from a remote FTP server to your agent computer.
- Upload ASCII, binary, or EBCDIC (i5/OS only) files from your agent computer to a remote FTP server.

Your agent administrator can set up the agent to run as an FTP client, FTP server, or both.

EBCDIC File Transfers

The EBCDIC transfer type applies to CA WA Agent for i5/OS only.

For the QSYS file system on i5/OS systems, you can only transfer members of FILE objects.

Note: For more information about FTP restrictions on i5/OS systems, see the IBM documentation.

Wildcard Characters in File Names

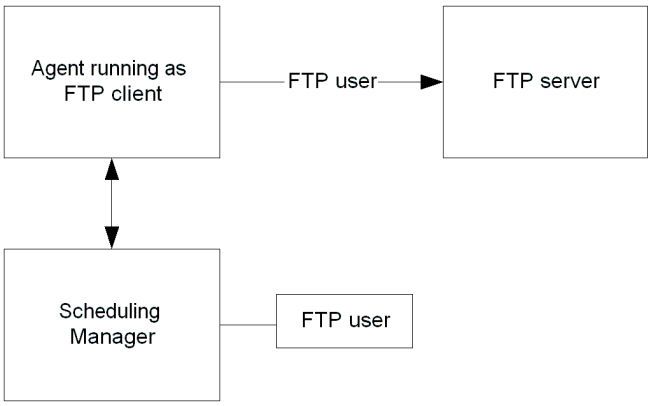
You can use wildcards in file names for ASCII, binary, and EBCDIC transfers. The asterisk (*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.

Note: Do not use wildcards in file names for auto-detect transfers. If you use wildcards in an auto-detect FTP job, the job will fail.

Running the Agent as an FTP Client

If the agent runs as an FTP client, the agent can log in to remote FTP servers and download files from and upload files to those servers.

The following diagram shows the relationship between an agent running as an FTP client, the scheduling manager, and an FTP server:



Note: The FTP user ID used to connect to the FTP server must be defined on the scheduling manager.

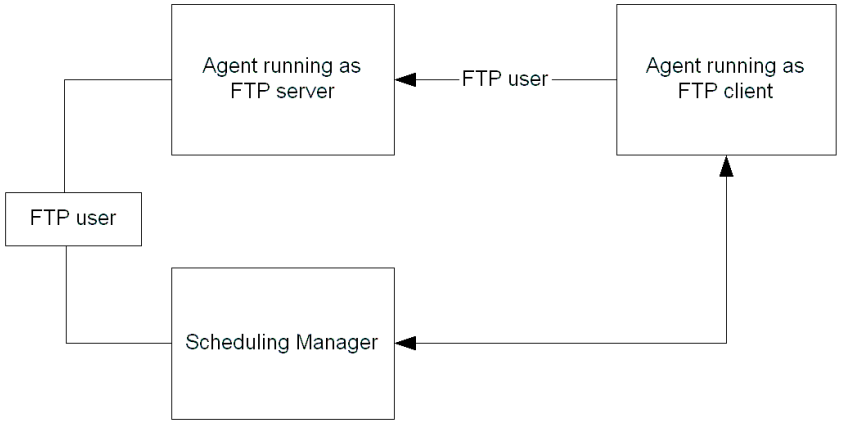
When the agent runs as an FTP client only, other FTP clients (such as other agents) cannot log in to the agent to transfer files. To let other FTP clients log in and transfer files, the agent administrator needs to set up the agent to run as an FTP server.

Running the Agent as an FTP Server

The agent supports a built-in FTP server capability. The agent administrator can enable the agent to act as a generic FTP server in addition to its other roles. This server adheres to the security rules established for the agent.

If the agent runs as an FTP server, clients can log in to the agent and transfer files.

The following diagram shows the relationship between an agent running as an FTP server, the scheduling manager, and another agent running as an FTP client:



Note: The FTP user ID used to connect to the agent running as an FTP server must be defined on that agent and the scheduling manager.

If the agent is configured as an FTP server, the agent can handle ASCII, binary, and EBCDIC file transfers, wildcard requests, simple GET and PUT requests for single files, and MGET and MPUT requests for multiple files.

The agent has a secure store of FTP server user IDs and associated passwords. The ftpusers.txt file, located in the directory where the agent is installed, stores these user IDs and their corresponding hashed passwords.

The agent running as an FTP server does not support anonymous FTP requests. For audit purposes, the agent provides a detailed log of all FTP requests.

Defining FTP Jobs

You can define an FTP (FTP_JOB) job to automate FTP transfers. The output is directed to the spool file through an FTP server.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Statements

To define an FTP job, you must specify the following statements:

- AGENT (see page 226)
- LOCALFILENAME (see page 407)
- REMOTEFILENAME (see page 530)
- SERVERADDR (see page 572)
- TRANSFERCODETYPE (see page 619)
- TRANSFERDIRECTION (see page 625)
- USER

Optional Statements

You can specify the following optional statements for an FTP job:

- JOBCLASS
- LOCALUSER (see page 412)
- SERVERPORT (see page 575)
- SITECMD (see page 583)

Example: Upload a Windows File to a Mainframe Data Set

This example uploads a file named TESTS to the mainframe. By default, the REMOTEFILENAME statement uses the user ID of the logged-on user for the data set high-level prefix.

Note: To specify a different high-level prefix, fully qualify the value of the REMOTEFILENAME statement and wrap double quotes around it.

```
AGENT WINAGENT
LOCALFILENAME 'C:\My Documents\Microsoft Office Directories+
WordPad\tests.txt'
REMOTEFILENAME ‘TESTS.DATA'
SERVERADDR MAINFRAME.COM
TRANSFERCODETYPE U
TRANSFERDIRECTION UPLOAD
USER causer
```

Example: Download a Mainframe Data Set to a Windows File

This example downloads the data set member CA07XX01 to a Windows computer. The value of the REMOTEFILENAME statement is fully qualified by wrapping double quotes around the single quotes.

```
AGENT WINAGENT
LOCALFILENAME 'C:\My Documents\File Transfers\ca07xx01.txt'
REMOTEFILENAME "'SYS4.CA7.JCLLIB(CA07XX01)'"
SERVERADDR MAINFRAME.COM
TRANSFERCODETYPE A
TRANSFERDIRECTION DOWNLOAD
USER lmla
```

Example: Download a File from a UNIX Computer to a Windows Computer

This example downloads a file from a UNIX server to a Windows computer. After the download is complete, the job completes.

```
AGENT SYSAGENT
USER test
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE A
REMOTEFILENAME /u1/test/ftpdata/textfile
LOCALFILENAME c:\test\textfile
```

Example: Download a Binary File on UNIX

The FTP job in this example uses a binary transfer.

```
AGENT UNIXAGENT
USER test
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE B
REMOTEFILENAME /u1/test/ftpdata/WAmgr
LOCALFILENAME /export/home/test/ftpdata/transf.bin
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5unix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

Transfer Files Using SSL FTP

If the FTP server supports SSL FTP and the agent FTP client has SSL FTP configured, you can securely transfer data using Secure Sockets Layer (SSL) communication.

Notes:

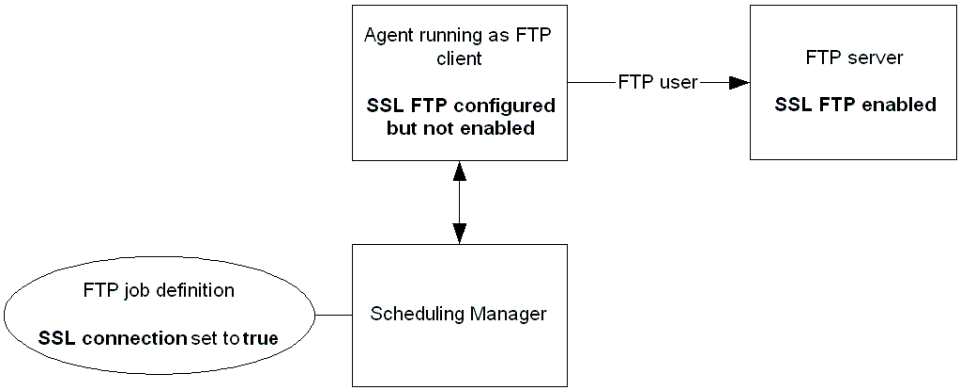
- To transfer data using SSL, the FTP server must have SSL FTP enabled and the FTP client must have SSL configured (SSL FTP can be enabled or disabled).
- If you do not specify the SSL operand, the data is transferred using the default FTP setting (regular FTP or SSL FTP) set on the agent FTP client as follows:
  - If the agent FTP client has SSL FTP enabled, all FTP jobs on that agent automatically use SSL FTP.

    Note: If the FTP server does not support SSL FTP, you must set the SSL operand in the TRANSFERCODETYPE statement to NO. Otherwise, the job will fail.
  - If the agent FTP client has SSL FTP disabled, all FTP jobs on that agent automatically use regular FTP.

To transfer files using SSL FTP, specify the SSL operand in the TRANSFERCODETYPE statement in the job definition.

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.

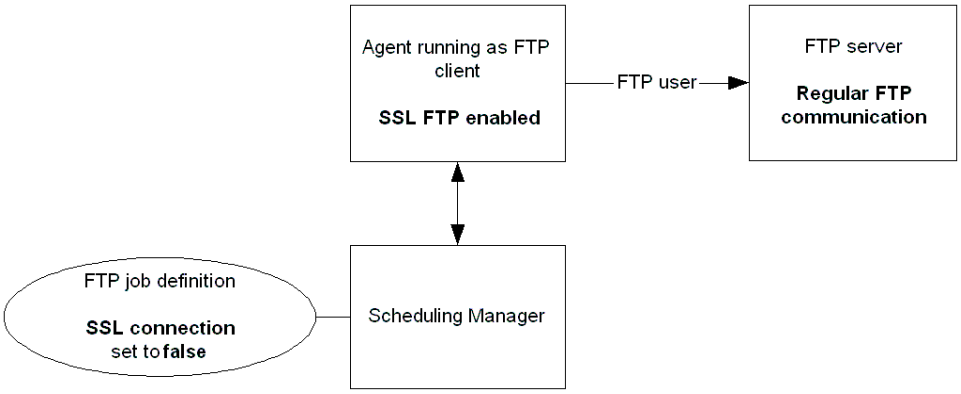


The following FTP job uploads the filename.txt file from the agent FTP client to the c:\uploaded_files directory on a remote Windows server. The SSL operand in the TRANSFERCODETYPE statement is set to YES to transfer the file securely. Although the agent FTP client does not have SSL FTP enabled, the 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).

```
AGENT WINAGENT
USER user1
SERVERADDR winserver
SERVERPORT 21
TRANSFERDIRECTION U
TRANSFERCODETYPE U SSL(YES)
REMOTEFILENAME c:\uploaded_files\filename.txt
LOCALFILENAME 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 c:\downloaded_files directory on the local computer. Because the FTP server does not support SSL FTP, the SSL operand in the TRANSFERCODETYPE statement is set to NO so the job does not fail.

```
AGENT WINAGENT
USER user1
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A SSL(NO)
REMOTEFILENAME /files_to_download/filename.txt
LOCALFILENAME c:\downloaded_files\filename.txt
```

Compress Data for FTP

When running FTP workload between an FTP client and FTP server that are both run on the agent software, you can compress the data for the transfer by specifying the compression level in the job definition.

Note: If the compression level is specified and the FTP server or the FTP client does not run on the agent, the data will be transferred without compression.

To compress the data that you want to transfer, specify the compression level using the COMPRESS operand in the TRANSFERCODETYPE statement. The compression level is a one-digit value from 0 to 9, where 0 is no data compression and 9 is the best data compression.

Example: Compress a File

In this example, the local computer has an agent running as an FTP client. The remote computer 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.

```
AGENT WINAGENT
USER user1
SERVERADDR aixunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A COMPRESS(3)
REMOTEFILENAME /files_to_download/largefile.txt
LOCALFILENAME c:\downloaded_files\largefile.txt
```

Send Site-Specific FTP Commands to FTP Servers

When you define an FTP job, you can specify one or more commands to execute prior to file transfer. You can use this feature to send site-specific FTP commands to FTP servers.

To send a site-specific FTP command to an FTP server, add the SITECMD statement in the job definition. You can specify multiple SITECMD statements in an FTP job.

Example: Send Two FTP Commands to an FTP Server

This example sends two FTP commands to the FTP server prior to transferring a file:

```
AGENT FTPAGENT
USER user1
SERVERADDR ftp.example.com
SERVERPORT 21
TRANSFERDIRECTION D
TRANSFERCODETYPE E COMPRESS(8)
REMOTEFILENAME /pub/cazip.exe
LOCALFILENAME /tmp/bla
SITECMD site date
SITECMD site recfm=FB
```

Verify the FTP Job Status

You can verify that the transfer completed successfully without file corruption.

To verify that the transfer completed successfully, check the job's spool file for the following responses:

- If the data was transferred using SSL FTP, the spool file contains the following response:

  ```
  AUTH TLS
  234 AUTH command OK. Initializing SSL connection.
  ```

- If the data was compressed and transferred without file corruption, the spool file contains a response as follows:

  ```
  Downloaded 81920/26119 bytes (original/compressed) in 0.161 seconds, 496.89 Kbytes/sec.
  ```

- If the file was downloaded successfully, the spool file contains the following response:

  ```
  Download successful
  ```

Chapter 6: HP Integrity NonStop Jobs

This section contains the following topics:

HP Integrity NonStop Jobs (see page 105)
Defining HP Integrity NonStop Jobs (see page 105)
How to Work with an HP Integrity NonStop Job (see page 106)

HP Integrity NonStop Jobs

The HP Integrity NonStop job lets you run workload on an HP Integrity NonStop system.

Note: To run HP Integrity NonStop workload, your system requires CA WA Agent for HP Integrity NonStop.

The CA WA Agent for HP Integrity NonStop r11.3.1 creates a spool file, job log file, output file, and error file.

For more details on HP Integrity NonStop jobs, see the following topics:

- Defining HP Integrity NonStop Jobs (see page 105)
- How to Work with an HP Integrity NonStop Job (see page 106)

Defining HP Integrity NonStop Jobs

An HP Integrity NonStop job definition consists of the required and optional statements covered in this topic.

Note: To run HP Integrity NonStop workload, your system requires CA WA Agent for HP Integrity NonStop.

Required Statements

You must code the following statements to define an HP Integrity NonStop job:

- AGENT (see page 226)
- COMMAND (see page 289)
- USER

Optional Statements

You can code the following statements if you want in an HP Integrity NonStop job definition:

- ARGS (see page 257)
- ASSIGN (see page 262)
- DEFINE (see page 309)
- ENVAR
- PARAMETER (see page 451)

Example: Define an HP Integrity NonStop Job

This example shows an HP Integrity NonStop job definition.

```
USER prod.glsys
AGENT PROAGENT
COMMAND $C35.PROD.GETDEF
ENVAR STDOUT=>>$C35.PROD.oabcout1
ENVAR STDERR=>>$c35.srkobj.err1
ENVAR DEF1="DABC;CLASS=MAP;FILE=$c35.PROD.GETDEF"
```

How to Work with an HP Integrity NonStop Job

This process shows you how to work with an HP Integrity NonStop job run by the CA WA Agent for HP Integrity NonStop.

Follow these steps:

1. Do the following:
   - Specify the agent name for an HP Integrity NonStop job (see page 107).
   - Specify the user ID that an HP Integrity NonStop job runs under (see page 107).
   - Run a command on the CA WA Agent for HP Integrity NonStop (see page 107).
2. Use any of the following procedures that apply:
   - Assign a Logical File Name to a Physical File on an HP Integrity NonStop System (see page 108)
   - Define a Logical Name for a File-System Element on an HP Integrity NonStop System (see page 108)
   - Pass Environment Variables to an HP Integrity NonStop Job (see page 108)
   - Pass a Parameter to an HP Integrity NonStop Job (see page 109)
   - Pass Positional Arguments to an HP Integrity NonStop Job (see page 109)

Specify the Agent Name for an HP Integrity NonStop Job

You must specify the name of the CA WA Agent for HP Integrity NonStop that will run the HP Integrity NonStop job.

To specify the agent name for an HP Integrity NonStop job, code the AGENT statement (see page 226) in the job definition or in the job parameters.

Note: An agent name specified in the job definition overrides an agent name specified in the job parameters.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Specify the User ID that an HP Integrity NonStop Job Runs Under

You must specify the user ID used to run the HP Integrity NonStop job.

To specify the user ID that the HP Integrity NonStop job runs under, code the USER statement in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Run a Command on the CA WA Agent for HP Integrity NonStop

You must specify the file containing the command that you want to run on the CA WA Agent for HP Integrity NonStop or directly specify the TACL command to run.

To run a command on the CA WA Agent for HP Integrity NonStop, code the COMMAND statement (see page 289) in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Assign a Logical File Name to a Physical File on an HP Integrity NonStop System

Rather than coding a physical file name in a program, you can assign a logical name to the file and code that name instead. You can also specify the characteristics of the physical file when you assign the logical name.

To assign a logical file name to a physical file on an HP Integrity NonStop system, code the ASSIGN statement (see page 262) in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Define a Logical Name for a File-System Element on an HP Integrity NonStop System

Rather than coding a file-system element name, you can assign a logical name to the element and code that name instead. For instance, you can define the logical name MYFIL1 to represent file $TEST.TST1.FINFOJOB. You can also specify the characteristics of the file-system element when you assign the logical name.

To define a logical name for a file-system element on an HP Integrity NonStop system, code the DEFINE statement (see page 309) in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Pass Environment Variables to an HP Integrity NonStop Job

You can pass environment variables to a batch file, script, command, or program being run on an HP Integrity NonStop system.

To pass environment variables to an HP Integrity NonStop job, code the ENVAR statement in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Pass a Parameter to an HP Integrity NonStop Job

You can specify a parameter to pass to a batch file, script, command, or program being run on an HP Integrity NonStop system. The parameter persists only for the current job and it is set in the environment of the command that the agent runs.

To pass a parameter to an HP Integrity NonStop job, code the PARAMETER Statement (see page 451) in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Pass Positional Arguments to an HP Integrity NonStop Job

You can pass a string of positional arguments to a batch file, script, command, or program being run on an HP Integrity NonStop system.

To pass positional arguments to an HP Integrity NonStop job, code the ARGS statement (see page 257) in the job definition.

More Information:

How to Work with an HP Integrity NonStop Job (see page 106)

Chapter 7: i5/OS Jobs

This section contains the following topics:

i5/OS Jobs (see page 111)
Defining i5/OS Jobs (see page 113)
Pass Positional Parameters (see page 114)
Use a User's Library List (see page 114)
Pass Keyword Parameters to SBMJOB (see page 114)
Set an i5/OS Job's Process Priority (see page 116)
Returning a Job's Exit Status to the Scheduling Manager (see page 116)
Specify Data for a Local Data Area (see page 119)

i5/OS Jobs

The i5/OS job lets you run a program or issue a command on an i5/OS system. You can run i5/OS jobs in the following file systems:

- Root file system
- Open systems file system (QOpenSys)
- Library file system (QSYS)

Note: To run these jobs, your system requires CA WA Agent for i5/OS.

You can specify the following details in an i5/OS job definition:

- Library name, library list, and current library for running a program
- The i5/OS job name, options under which the job will run, where it will run, and which user will run it
- Ending exit value of the program, such as a severity code

You can define parameter values that you want to pass to a program at the time the program is invoked.

Note: Default values may be set for certain parameters, such as the i5/OS user ID that the jobs run under. Contact your agent administrator about the parameters set in the agentparm.txt file.

Running UNIX Workload on a System i5 Computer

You can schedule most UNIX workload, such as UNIX scripts, in the PASE environment on the i5/OS operating system.

Note: For more information about UNIX workload that can run in the PASE environment, see the IBM i5/OS documentation.

i5/OS Naming Conventions

When specifying i5/OS paths and names in your workload, you can use the following naming conventions, depending on where the file is located on the i5/OS system:

- Root file system

  To specify a file in the root file system, use UNIX path and file formats.

- Open systems file system (QOpenSys)

  To specify a file in QOpenSys, use UNIX path and file formats. QOpenSys file names are case-sensitive.

- Library file system (QSYS)

  To specify an object in QSYS, use one of the following formats (unless described differently in the job definition syntax):

  - Path format

    `/QSYS.LIB/`*`library`*`.LIB/`*`object`*`.`*`type`*`/`

    To specify *FILE objects, use the following format:

    `/QSYS.LIB/`*`library`*`.LIB/`*`object`*`.FILE/`*`member`*`.MBR`

  - i5/OS standard format

    `library/object/type`

    To specify *FILE objects, use the following format:

    `library/object``/*FILE(`*`member`*`)`

    Note: *FILE is optional when *member* is specified. That is, you can specify a file member using the following format:

    `library/object``(`*`member`*`)`

Notes:

- The values for *library*, *object*, *type*, and *member* can be up to 10 characters each.
- You can use *ALL to match any name.
- You can use *FIRST for *member*.
- You can use generic names for *library* and *object*.

Defining i5/OS Jobs

You can define an i5/OS (AS400_JOB) 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 Statements

To define an i5/OS job, you must specify the following statements:

- AGENT (see page 226)
- AS400FILE (see page 259), CLPNAME (see page 282), or COMMAND (see page 289)

Optional Statements

You can specify the following optional statements for an i5/OS job:

- AS400LIB (see page 261)
- CCEXIT (see page 276)
- CURLIB (see page 305)
- EXITCODE
- JOBCLASS
- JOBD (see page 396)
- JOBNAME (see page 397)
- JOBQ (see page 401)
- LDA (see page 403)
- LIBL (see page 404)
- OTHERS (see page 436)
- PARAM (see page 445)
- PROCESS_PRIORITY (see page 484)
- USER

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.

```
AGENT I5AGENT
CLPNAME MFGDATA
AS400LIB PRODJOBS
```

Pass Positional Parameters

When running workload, you might need to pass data between jobs and across platforms. You can pass positional parameters to an i5/OS program in your job definition. Positional parameters are variables that can be passed to a program at the time the program is invoked. The parameters are assigned in the order they are passed.

To pass positional parameters to an i5/OS program, specify the PARAM statement in the job definition. You can specify multiple PARAM statements within a single job definition.

Example: Pass Multiple Parameters to an i5/OS Job

This example passes eight parameters to an i5/OS program. Each parameter is enclosed with single quotation marks and separated with a blank space.

```
PARAM 'PAYROLL' '*LIBL' 'ABC' '01' '1' 'P' 'TAP02' '0'
```

Use a User's Library List

The agent uses the library list in the job's job description by default. However, if the user is defined, you can set up your job definition to use the user's library list instead.

To use a user's library list, do one of the following:

- Specify the following statement in the job definition:

  ```
  CURLIB *USRPRF
  ```

  When the job runs, it uses the user's current library.

- Specify the following statements in the job definition:

  ```
  JOBD *USRPRF
  LIBL *JOBD
  ```

  When the job runs, it accesses the user's library list by using the job description assigned to the user.

Pass Keyword Parameters to SBMJOB

When the scheduling manager submits a job to the i5/OS system, the job must pass through the SBMJOB command to execute. The following statements map to parameters for the SBMJOB command.

Statements	SBMJOB Parameters
USER	USER

Statements	SBMJOB Parameters
JOBD	JOBD
LIBL	INLLIBL
JOBQ	JOBQ
CURLIB	CURLIB
JOBNAME	JOB

You can also pass additional keyword parameters, such as OUTQ(*JOBD), to the SBMJOB command.

To pass an SBMJOB command keyword and value combination, specify the OTHERS statement in the job definition.

Note: The special value for these SBMJOB parameters, such as *USRPRF and *JOBD, also apply to the statements. You can use these special values in your job definitions. For more information about the SBMJOB parameters and their special values, see the IBM documentation.

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.

```
AGENT I5AGENT
CLPNAME PAYJOB
OTHERS PRTDEV(*JOBD) OUTQ(*JOBD)
```

Set an i5/OS Job's Process Priority

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.

To set an i5/OS job's process priority, specify the priority level using the PROCESS_PRIORITY statement in the job definition.

Example: Set the Priority for an i5/OS Job

In this example, the process priority for an i5/OS job is set to ABOVE_NORMAL.

```
AGENT I5AG
CLPNAME PAYROLL
PROCESS_PRIORITY ABOVE_NORMAL
```

If you do not specify the process priority in the job definition, the job's process priority is set to NORMAL by default.

Returning a Job's Exit Status to the Scheduling Manager

A job's exit code indicates whether the job completed successfully or failed. By default, the agent sends the job's ending severity code to the scheduling manager when a job completes. By default, an exit code of zero (0) is interpreted as job success and any other number indicates job failure.

In addition to sending the job's ending severity code, there are other ways to return a job's exit status. For example, you can send the return code of an ILE program or module as the exit status or you can specify a user-defined exit code of 100 as success.

You can return a job's exit status to the scheduling manager using the following methods:

- Send a program's return code using the CCEXIT statement
- Send a user-defined exit code using the EXITCODE statement

Send a Program's Return Code

When a job completes, the agent sends the job's exit code to the scheduling manager. By default, the agent sends the job's ending severity code as the job's exit code.

Instead of sending the job's ending severity code, the agent can send the return code of one of the following:

- An ILE program or module
- AN OPM program

For example, if your job runs an ILE C, ILE RPG, OPM RPG, or OPM Cobol program that contains an exit or return statement, the agent sends that return code as the exit code.

To send a program's return code instead of the job's ending severity code, specify the CCEXIT statement with the value *USER (for an ILE program or module) or *PROGRAM (for an OPM program) in your job definition.

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 the scheduling manager.

```
AGENT I5AG
CLPNAME PAYROLL
CCEXIT *PROGRAM
```

Example: Send an ILE 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 the scheduling manager. Ending severity codes of 0 (zero) or 10 indicate job success.

```
AGENT I5AG
CLPNAME SALARY
CCEXIT *USER
EXITCODE 10 SUCCESS
EXITCODE 0 SUCCESS
```

Note: The i5 system always writes the job's ending severity code to the job's spool file. You can check the spool file for the job's ending severity code for more information about the job status. For example, suppose that the C program's return code indicates failure, but the ending severity code shown in the spool file is 10, which might indicate that the job ran with a minor issue. Assuming that this issue can be ignored, you can indicate ending severity codes of 10 as job success using the EXITCODE statement.

Send a User-Defined Exit Code

By default, an exit code of zero (0) is interpreted as job success and any other number indicates job failure. However, you can map exit codes other than 0 as job success.

To send a user-defined exit code, specify the EXITCODE statement in the job definition.

Example: Send Exit Code 100 as Success

This example runs the PAYPROG program. The program is considered to have completed successfully if it returns an exit code of 0 or 100.

```
AGENT I5AG
CLPNAME PAYPROG
EXITCODE 100 SUCCESS
EXITCODE 0 SUCCESS
```

Example: Send Exit Code 40 as Failure

This example runs the RECPROG program. The program is considered to have failed if it returns an exit code of 40. All other exit codes in the range from 0 to 255 indicate job success.

```
AGENT I5AG
CLPNAME RECPROG
CCEXIT *PROGRAM
EXITCODE 40 FAILURE
EXITCODE 0-255 SUCCESS
```

Specify Data for a Local Data Area

The local data area is a temporary 1024-byte storage area that exists for the duration of an i5/OS 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. When the job is submitted, the agent initializes the local data area with the specified data. When the job completes, the local data area is destroyed automatically by the operating system.

To specify data for a local data area, add the LDA statement in the job definition.

Example: Specify Data for the Local Data Area in Hexadecimal Format

In the following example, when the job is submitted, the agent initializes the local data area with hexadecimal data. When the job completes, the local data area is destroyed automatically by the operating system.

```
AGENT A03TS1
CLPNAME IVP
LDA X'C1C2C3C4'
```

Chapter 8: Monitoring Jobs

This section contains the following topics:

Monitoring Jobs (see page 121)
Defining CPU Monitoring Jobs (see page 122)
Defining Disk Monitoring Jobs (see page 123)
Defining IP Monitoring Jobs (see page 125)
Defining Process Monitoring Jobs (see page 126)
Defining Text File Reading and Monitoring Jobs (see page 127)
Defining Windows Event Log Monitoring Jobs (see page 130)
Defining Windows Service Monitoring Jobs (see page 132)

Monitoring Jobs

Monitoring jobs let you monitor different aspects of your system.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

You can define the following monitoring jobs:

CPU Monitoring

Lets you monitor CPU usage.

Disk Monitoring

Lets you monitor disk space.

IP Monitoring

Lets you monitor an IP address.

Process Monitoring

Lets you monitor process execution.

Text File Reading and Monitoring

Lets you search a text file for a string.

Windows Event Log Monitoring

Lets you monitor a Windows event log.

Windows Service Monitoring

Lets you monitor the status of Windows services.

Defining CPU Monitoring Jobs

You can define a CPU Monitoring (CPU_MON) 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 Statements

To define a CPU Monitoring job, you must specify the following statement:

- AGENT (see page 226)

Optional Statements

You can specify the following optional statements for a CPU Monitoring job:

- CPU (see page 299)
- JOBCLASS
- USER
- WAITMODE (see page 642)

Example: Monitor Used CPU

This example continuously monitors the CPU used on the SYSAGENT computer. Each time the used CPU exceeds 75 percent, an alert named CPU is triggered. The job continues monitoring the CPU usage until it is ended manually.

```
AGENT SYSAGENT
CPU FROM(75) TO(100) CONTINUOUS(CPU) USED
```

Defining Disk Monitoring Jobs

On Windows and UNIX systems, you can define a Disk Monitoring (DISK_MON) 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 Statements

To define a Disk Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- DISK (see page 319)

Optional Statements

You can specify the following optional statements for a Disk Monitoring job:

- JOBCLASS
- USER
- WAITMODE (see page 642)

Example: Monitor Available Space on a Windows Drive

This example monitors the C drive on a Windows computer for available space. The job completes immediately and reports the available space in megabytes.

```
AGENT SYSAGENT
DISK C FORMAT(MB)
```

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 named DISK is issued. The job continues monitoring the disk space until it is ended manually.

```
AGENT SYSAGENT
DISK /export/home FORMAT(PERCENT) FROM(90) USED CONTINUOUS(DISK)
```

Example: Continuously Monitor Used Space on a Windows Drive

This example continuously monitors the C drive on a Windows computer for used space. Each time the used disk space falls below 16 percent or exceeds 95 percent, an alert named DISK is triggered. The job continues monitoring the disk space until it is ended manually.

```
AGENT SYSAGENT
DISK C FORMAT(PERCENT) FROM(16) TO(95) USED OUTSIDE CONTINUOUS(DISK)
```

Example: Monitor the System Auxiliary Storage Pool on an i5/OS System

This example monitors the system auxiliary storage pool for available disk space. The job completes immediately and reports the available space in megabytes.

Note: The system ASP is denoted by the forward slash (/). You can also specify the system ASP as /dev/QASP01.

```
AGENT I5AGENT
DISK / FORMAT(MB)
```

Example: Monitor an Auxiliary Storage Pool Using a Range

This example monitors the auxiliary storage spool for used disk space. When the used disk space falls between 90 and 100 percent, the job completes.

```
AGENT I5AGENT
DISK /dev/QASP02 FORMAT(PERCENT) FROM(90) TO(100) USED
```

Example: Monitor a Mounted File System for a Storage Space Value Outside of a Range

This example monitors an NFS server mounted at /ul for used disk space. When the used disk space falls below 16 percent or exceeds 95 percent, the job completes.

```
AGENT I5AGENT
DISK /u1 FORMAT(PERCENT) FROM(16) TO(95) USED OUTSIDE
```

Defining IP Monitoring Jobs

You can define an IP Monitoring (IP_MON) 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 Statements

To define an IP Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- IPADDRESS (see page 386)
- STATUS (see page 593)

Optional Statements

You can specify the following optional statements for an IP Monitoring job:

- IPPORT (see page 386)
- JOBCLASS
- USER

Example: Monitor an IP Address and Port

This example monitors a device at IP address 172.24.2.20 and port 9401. When the job runs, it immediately checks if the device is running. If the device is running, the job completes. If the device is not running, the job fails.

```
AGENT SYSAGENT
IPADDRESS 172.24.2.20
IPPORT 9401
STATUS RUNNING NOW
```

Defining Process Monitoring Jobs

You can define a Process Monitoring (PROCESS_MON) 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 Statements

To define a Process Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- PROCESS (see page 480)
- STATUS (see page 593)

Optional Statements

You can specify the following optional statements for a Process Monitoring job:

- JOBCLASS
- USER

Example: Monitor the Agent Process on Windows Using a Full Path

Suppose that the WINAGENT computer runs multiple agents. To monitor the cybAgent.exe process, the full path to that process is specified. The job checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails. The process name is enclosed in single quotes because it contains a space.

```
AGENT WINAGENT
PROCESS 'c:\Program files\agentdir\cybAgent.exe'
STATUS RUNNING NOW
```

Example: Monitor the Agent Process on UNIX Using a Process Name

This example monitors the cybAgent process on a UNIX computer. The job checks the process status immediately and completes successfully if the process is running.

Note: You can also specify a PID number in place of the process name.

```
AGENT SYSAGENT
PROCESS cybAgent
STATUS RUNNING NOW
```

Example: Monitor a Process That Has Any Job Number and Is Running Under Any User Name

This example monitors for processes named CYBAGENT running on an i5/OS computer, 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 single quotation marks so that the text following /* is not interpreted as a comment.

```
AGENT I5AGENT
PROCESS '*ALL/*ALL/CYBAGENT'
STATUS RUNNING 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.

```
AGENT I5AGENT
PROCESS *ALL/JDOE/CALC*
STATUS STOPPED WAIT
```

Example: Monitor a UNIX Process on an i5/OS Operating System

This example monitors the BMPROC process in the PASE environment on the i5/OS operating system. The job checks the process status immediately and completes successfully if the process is running.

```
AGENT I5AGENT
PROCESS BMPROC
STATUS RUNNING NOW
```

Note: You can monitor a UNIX process on an i5/OS operating system by specifying the process name or process ID. However, to monitor an i5/OS process, you must specify the three-part process ID (*jobnumber*/*username*/*jobname*).

Defining Text File Reading and Monitoring Jobs

You can define a Text File Reading and Monitoring (TEXT_MON) 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 Statements

To define a Text File Reading and Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- SEARCHRANGE (see page 566)
- TEXTFILE (see page 606)
- TEXTSTRING (see page 609)

Optional Statements

You can specify the following optional statements for a Text File Reading and Monitoring job:

- ENCODING (see page 335)
- JOBCLASS
- TIMEFORMAT (see page 613)
- USER
- WAITMODE (see page 642)

Example: Search for a Text String on a Windows System

This example searches for a text string in a text file on a Windows computer. The path to the text file is enclosed in single quotation marks because the path contains a space. The job searches between lines 1 and 20. The job completes successfully if the string is found.

```
AGENT SYSAGENT
TEXTFILE 'c:\Program Files\CA\component.txt'
TEXTSTRING 'CreateService failed' EXIST
SEARCHRANGE LINE FROM(1) TO(20)
```

Example: Monitor an i5/OS Text File for a String in the First 20 Lines of the File

This example searches for a text string in the DATA member of a QSYS file object on an i5/OS computer. The job searches the content between lines 1 and 20. The job completes successfully if the string is found.

```
AGENT I5AGENT
TEXTFILE /QSYS.LIB/LIBRARY.LIB/RESULTS.FILE/DATA.MBR
TEXTSTRING 'Create file failed' EXIST
SEARCHRANGE LINE FROM(1) TO(20)
```

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 operand REGEX in the SEARCHRANGE statement.

```
AGENT SYSAGENT
TEXTFILE '/home/cybesp/file.txt'
TEXTSTRING '^\w{4,10}\.' EXIST
SEARCHRANGE REGEX FROM(\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 following syntax:

```
TEXTSTRING '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.

Monitor a File with Data Encoded in a non-ASCII Character Set

When you define a Text File Reading and Monitoring job, you can specify the character encoding of the text file to be monitored. For example, you can monitor a text file that contains data encoded in ISO-8859-1 (ISO Latin Alphabet No. 1 or ISO-LATIN-1).

To monitor a file with data encoded in a non-ASCII character set, add the ENCODING statement in the job definition.

Example: Monitor a Text File Encoded in ISO-8859-1

The following job monitors a text file that contains data encoded in ISO-8859-1 (ISO Latin Alphabet No. 1 or ISO-LATIN-1). The job completes successfully if the specified string is found.

```
AGENT MONAGT
TEXTFILE /export/home/logs/transactions.log
TEXTSTRING 'ERROR MESSAGE'
SEARCHRANGE LINE FROM(1) TO(50)
ENCODING ISO-8859-1
```

Defining Windows Event Log Monitoring Jobs

You can define a Windows Event Log Monitoring (EVENTLOG_MON) 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.

Note: To run these jobs, your system requires CA WA Agent for Windows.

Windows operating systems record events in different types of logs, including the following:

Application log

The application log contains events logged by applications or programs. For example, a database program might record a file error in the application log.

System log

The system log contains events logged by the Windows 2000 system components. For example, the failure of a driver or other system component to load during startup is recorded in the system log.

Security log

The security log can record security events such as valid and invalid logon attempts, as well as events related to resource use, such as creating, opening, or deleting files.

For more information on Windows logs, select Start, Settings, Control Panel, Administrative Tools, Event Viewer. Select any of the three log categories and double-click to view its property page.

Required Statements

To define a Windows Event Log Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- EVENTLOG (see page 351)

Optional Statements

You can specify the following optional statements for a Windows Event Log Monitoring job:

- EVENTCATEGORY (see page 347)
- EVENTCOMPUTER (see page 348)
- EVENTDESCRIPTION (see page 349)
- EVENTID (see page 350)
- EVENTSOURCE (see page 352)
- EVENTTIME (see page 353)
- EVENTTYPE (see page 354)
- JOBCLASS
- USER
- WAITMODE (see page 642)

Example: Monitor an Application Log that Occurs on or After a Specified Date and Time

This example monitors an application log that occurs on or after a specified date and time. When the job finds an application log that occurs any time on or after 6:30 a.m. on December 11, 2010, the job completes successfully.

```
AGENT SYSAGENT
EVENTLOG Application
EVENTTYPE info
EVENTCATEGORY None
EVENTSOURCE LLDSAPNT223
EVENTTIME FROM('06:30:00AM dec 11 2010')
```

Example: Continuously Monitor an Application Event Log

This example continuously monitors an application event log. An alert named ELOG is triggered for all instances of the INFO event type, where the event source is LLDSAPNT223, the event description contains the word started, and the event ID is less than or equal to 4000.

```
AGENT SYSAGENT
EVENTLOG Application CONTINUOUS(ELOG)
EVENTTYPE INFO
EVENTCATEGORY None
EVENTSOURCE LLDSAPNT223
EVENTDESCRIPTION started
EVENTID LE 4000
```

Defining Windows Service Monitoring Jobs

You can define a Windows Service Monitoring (SERVICE_MON) job to monitor services on a local Windows computer.

Note: To run these jobs, your system requires CA WA Agent for Windows.

Required Statements

To define a Windows Service Monitoring job, you must specify the following statements:

- AGENT (see page 226)
- SERVICENAME (see page 578)
- STATUS (see page 593)

Optional Statements

You can specify the following optional statements for a Windows Service Monitoring job:

- JOBCLASS
- USER

Example: Monitor a Running Service

This example monitors a Windows service named Proc Server. If the service is running, the job completes successfully. If the service is not running, the job continues monitoring it until the service starts running.

```
AGENT WINAGENT
SERVICENAME 'Proc Server'
STATUS RUNNING WAIT
```

Chapter 9: Oracle E-Business Suite Jobs

This section contains the following topics:

Oracle E-Business Suite Jobs (see page 135)
Defining Oracle E-Business Suite Copy Jobs (see page 136)
Defining Oracle E-Business Suite Request Set Jobs (see page 137)
Defining Oracle E-Business Suite Single Request Jobs (see page 142)

Oracle E-Business Suite Jobs

You can define jobs to run Oracle E-Business Suite workload.

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.

You can define the following Oracle E-Business Suite jobs:

Oracle E-Business Suite Copy Job

Copies an existing single request defined on Oracle Applications and runs it under the agent.

Oracle E-Business Suite Request Set

Runs multiple programs in an Oracle Applications application.

Oracle E-Business Suite Single Request

Runs a single program in an Oracle Applications application.

Defining Oracle E-Business Suite Copy Jobs

You can define an Oracle E-Business Suite Copy Job (OAC_JOB) to copy an existing single request defined on Oracle Applications 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 Statements

To define an Oracle E-Business Suite Copy Job, you must specify the following statements:

- AGENT (see page 226)
- REQUESTID (see page 537)

Optional Statements

You can specify the following optional statements for an Oracle E-Business Suite Job:

- CHILDMONITOR (see page 279)
- JOBCLASS
- MONITORDELAY (see page 427)
- OAUSER (see page 432)
- RESPNAME (see page 539)

Example: Copy a Single Request

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

Defining Oracle E-Business Suite Request Set Jobs

You can define an Oracle E-Business Suite Request Set (OA_JOB) job to run multiple programs in an Oracle Applications application.

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.

To define an Oracle E-Business Suite Request Set job, you require the following information from the original request set:

- Display name or short name of the application the request set belongs to
- Request set name
- User name the job runs under
- Responsibility name

Required Statements

To define an Oracle E-Business Suite Request Set job, you must specify the following statements:

- AGENT (see page 226)
- APPLDISPLNAME (see page 227) or APPLSHORTNAME (see page 228)
- REQUESTSET (see page 538) or REQSETDISPL (see page 536)

Notes:

- All Oracle E-Business Suite Request Set jobs require a user name. You must specify the OAUSER statement if a default user name is not specified in the oa.default.user parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- All Oracle E-Business Suite Request Set jobs require a responsibility name. You must specify the RESPNAME statement if a default responsibility name is not specified in the oa.default.responsibility parameter in the agent's agentparm.txt file. Otherwise, the job fails.

Optional Statements

You can specify the following optional statements for an Oracle E-Business Suite Request Set job:

- ARGDEFAULTS (see page 251)
- CHILDMONITOR (see page 279)
- CUSTOMPROP (see page 306)
- JOBCLASS
- MONITORDELAY (see page 427)

- OAUSER (see page 432)
- OUTPUTFORMAT (see page 444)
- PRINTCOPIES (see page 460)
- PRINTER (see page 467)
- PRINTSTYLE (see page 478)
- PROGARGS (see page 490)
- PROGDATA (see page 492)
- PROGNOTIFYDUSERS (see page 494)
- PROGNOTIFYUSERS (see page 496)
- PROGOUTPUTFORMAT (see page 497)
- PROGPRINTCOPIES (see page 498)
- PROGPRINTER (see page 500)
- PROGPRINTSTYLE (see page 501)
- PROGQUOTEDEFAULT (see page 503)
- PROGSAVEOUTPUT (see page 506)
- PROGTEMPLATELANG (see page 507)
- PROGTEMPLATETERR (see page 508)
- QUOTEDEFAULT (see page 519)
- RESPNAME (see page 539)
- SAVEOUTPUT (see page 562)
- TEMPLATELANG (see page 604)
- TEMPLATETERR (see page 605)
- USESETDEFAULTS (see page 640)

Example: Run a Request Set

In this example, the job runs the request set DATA10 on the Oracle Applications system.

```
AGENT CYBOA
REQUESTSET DATA10
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

Specify Data for an Individual Program in a Request Set

When you define an Oracle E-Business Suite Request Set job, you can specify data for an individual program in the request set. This data overrides the arguments and print parameters specified for the entire request set. You can specify the following data for a program:

- Program arguments
- Printer
- Print style
- Number of copies to print
- Whether to save the output from the program
- Whether to quote resolved expressions in default values
- Template language
- Template territory
- Output format
- Users to notify

To specify data for an individual program in a request set

1. Specify the order number of the program in the PROGDATA statement. The order number identifies the program in the request set.

   Note: If you do not specify an order number, the first program is treated as number one, and subsequent programs are treated as having the previous program number plus one.

2. Specify one or more of the following statements after the PROGDATA statement:
   - PROGARGS
   - PROGSAVEOUTPUT
   - PROGPRINTER
   - PROGPRINTSTYLE
   - PROGPRINTCOPIES
   - PROGQUOTEDEFAULT
   - PROGTEMPLATELANG
   - PROGTEMPLATETERR
   - PROGOUTPUTFORMAT
   - PROGNOTIFYUSERS
   - PROGNOTIFYDUSERS

These statements override the arguments and parameters specified for the entire request set.

Example: Specify Values for Individual Programs in a Request Set

Suppose that you want to define a Oracle E-Business Suite Request Set job to run a request set named EXTRACTS on the OAAGENT agent. The request set belongs to the application with the display name Application Object Library. 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 Q8.

The first PROGDATA statement identifies the first program in the request set. Subsequent statements override the default values for this program. The first program uses the arguments T23 and R1 and prints two copies using the Q1 printer in PORTRAIT style.

The second PROGDATA statement identifies the fifth program in the request set. Subsequent statements override the default values for this program. The fifth program uses arguments R and R1 and prints three copies using the Q2 printer in PORTRAIT style.

The other programs in the request set use the default values.

```
AGENT OAAGENT
REQUESTSET EXTRACTS
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PRINTCOPIES 1
PRINTSTYLE LANDSCAPE
PRINTER Q8
PROGDATA
   PROGARGS T23,,R1
   PROGPRINTER Q1
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 2
   PROGSAVEOUTPUT Y
PROGDATA 5
   PROGARGS R,R1,
   PROGPRINTER Q2
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 3
   PROGSAVEOUTPUT N
```

Specify Argument Values to Pass to a Program in a Request Set

You can pass argument values to an individual program in an Oracle E-Business Suite Request Set job. The job can also use the default values that are defined by the registered Concurrent Manager program. When an argument value is defined in both the job definition and as a default, the argument value in the job definition overrides the default.

To specify argument values to a program in a request set, specify the following statements in the job definition:

- (Optional) ARGDEFAULTS
- PROGARGS

  Note: Place the PROGARGS statement after the PROGDATA statement representing the program you want to pass argument values to.

Example: Specify Argument Values for a Program in a Request Set

This example runs an Oracle E-Business Suite Request Set job that uses the argument defaults in Oracle Applications and the argument values defined in the job definition. The second program in the request set has four arguments and you want to pass T23 as the first value and R1 as the fourth value. The argument string specifies placeholders for the second and third arguments, so the job uses the default values for those arguments.

```
AGENT OAAGENT
REQUESTSET EXTRACTS
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGSDEFAULTS Y
PROGDATA 2
   PROGARGS T23,,,,R1
   PROGPRINTER Q1
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 2
   PROGSAVEOUTPUT Y
```

Defining Oracle E-Business Suite Single Request Jobs

You can define an Oracle E-Business Suite Single Request (OA_JOB) job to run a single program in an Oracle Applications application.

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.

To define an Oracle E-Business Suite Single Request job, you require the following information from the original single request:

- Display name or short name of the application the single request belongs to
- Program name
- User name the job runs under
- Responsibility name

Required Statements

To define an Oracle E-Business Suite Single Request job, you must specify the following statements:

- AGENT (see page 226)
- APPLDISPLNAME (see page 227) or APPLSHORTNAME (see page 228)
- PROGRAM (see page 504) or PROGRAMDISPL (see page 505)

Notes:

- All Oracle E-Business Suite Single Request jobs require a user name. You must specify the OAUSER statement if a default user name is not specified in the oa.default.user parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- All Oracle E-Business Suite Single Request jobs require a responsibility name. You must specify the RESPNAME statement if a default responsibility name is not specified in the oa.default.responsibility parameter in the agent's agentparm.txt file. Otherwise, the job fails.

Optional Statements

You can specify the following optional statements for an Oracle E-Business Suite Single Request job:

- ARGDEFAULTS (see page 251)
- ARGS (see page 255)
- CHILDMONITOR (see page 279)
- CUSTOMPROP (see page 306)
- DESCRIPTION (see page 312)

- JOBCLASS
- MONITORDELAY (see page 427)
- NOTIFYDUSERS (see page 430)
- NOTIFYUSERS (see page 431)
- OAUSER (see page 432)
- OUTPUTFORMAT (see page 444)
- PRINTCOPIES (see page 460)
- PRINTER (see page 467)
- PRINTSTYLE (see page 478)
- QUOTEDEFAULT (see page 519)
- RESPNAME (see page 539)
- SAVEOUTPUT (see page 562)
- TEMPLATELANG (see page 604)
- TEMPLATETERR (see page 605)

Example: Run a Single Request

In this example, the job runs the single request XXCOFI on the Oracle Applications system.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

Specify Argument Values to Pass to a Program in a Single Request

You can pass argument values to a program in an Oracle E-Business Suite Single Request job. The job can also use the default values that are defined by the registered Concurrent Manager program. When an argument value is defined in both the job definition and as a default, the argument value in the job definition overrides the default.

To specify argument values to pass to a program in a single request, specify the following statements in the job definition:

- ARGS
- (Optional) ARGDEFAULTS

Example: Specify Argument Values for a Single Request

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 Applications 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 Applications.
- The fourth argument is not specified in the job definition or defined as a default on Oracle Applications, so the agent passes an empty string for that argument.

```
AGENT OAAGENT
PROGRAM FNDSCARU
APPLDISPLNAME 'Application Object Library'
ARGS T,,X23,,
ARGSDEFAULTS Y
```

Chapter 10: PeopleSoft Jobs

This section contains the following topics:

PeopleSoft Jobs (see page 145)
Defining PeopleSoft Jobs (see page 146)
Distribute a PeopleSoft Report (see page 148)
Send the Output of a PeopleSoft Job to a File (see page 149)
Send the Output of a PeopleSoft Job to a Printer (see page 150)

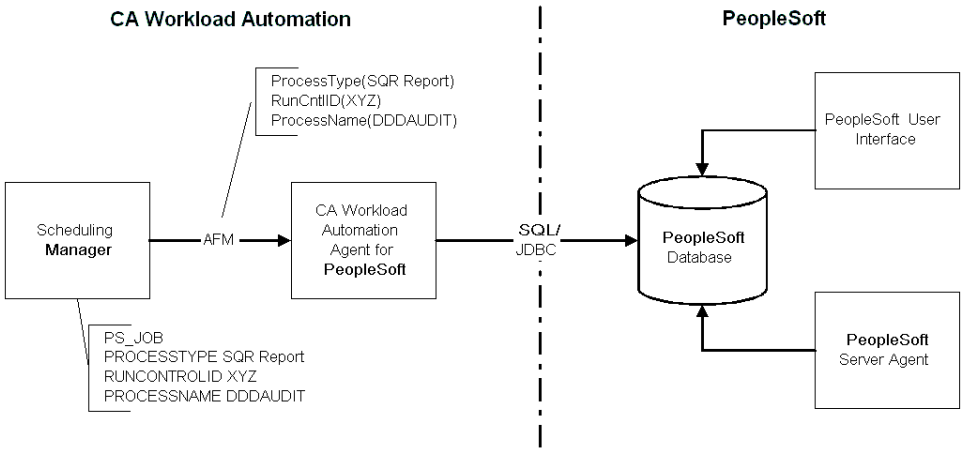
PeopleSoft Jobs

PeopleSoft jobs let you run different types of PeopleSoft processes defined in your PeopleSoft system. For example, you can define PeopleSoft jobs to execute PeopleSoft programs and report the program status.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for PeopleSoft.

When you define a PeopleSoft job, 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 also pass run control parameter values that will be stored in the corresponding run control table.

When a PeopleSoft program runs, it modifies its run status (RUNSTATUS) in the PSPRCSRQST table in the PS database. The following diagram shows the functional relationship between the scheduling manager, the agent, and the PeopleSoft system:



Defining PeopleSoft Jobs

You can define a PeopleSoft (PS_JOB) job to schedule workload to run in PeopleSoft. The job runs a PeopleSoft request or a collection of requests.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for PeopleSoft.

Required Statements

To define a PeopleSoft job, you must specify the following statements:

- AGENT (see page 226)
- PROCESSNAME (see page 488)
- PROCESSTYPE (see page 489)

Notes:

- All PeopleSoft jobs require an operator ID. You must specify the PSOPRID statement if a default operator ID is not specified in the ps.default.oprId parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- All PeopleSoft jobs require a run control ID. You must specify the RUNCONTROLID statement if a default run control ID is not specified in the ps.default.runCntlId parameter in the agent's agentparm.txt file. Otherwise, the job fails.

Optional Statements

You can specify the following optional statements for a PeopleSoft job:

- ARGS (see page 256)
- DISABLE_RESTART (see page 318)
- DISTRFOLDER (see page 324)
- DISTRLISTROLES (see page 325)
- DISTRLISTUSERS (see page 327)
- EMAILADDR (see page 329)
- EMAILLOG (see page 331)
- EMAILSUBJECT (see page 332)
- EMAILTEXT (see page 333)
- EMAILWEBREPORT (see page 334)
- ENVAR
- JOBCLASS
- OUTDESTFORMAT (see page 437)

- OUTDESTPATH (see page 439)
- OUTDESTTYPE (see page 441)
- PSOPRID (see page 517)
- RUNCONTROLARGS (see page 547)
- RUNCONTROLID (see page 549)
- RUNCONTROLTABLE (see page 550)
- SERVERNAME (see page 573)
- SKIPPARMUPDATES (see page 584)
- TIMEZONE (see page 617)
- USER

Example: Run a PeopleSoft Report

This example runs an Application Engine process named PSQUERY. The job runs on the agent named PS_NY.

```
AGENT PS_NY
PROCESSNAME PSQUERY
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

Distribute a PeopleSoft Report

If you specify EMAIL or WEB as the output destination type (OUTDESTTYPE), you can distribute a PeopleSoft report electronically to operators, groups of people, or individuals.

To distribute a PeopleSoft report

1. Add the OUTDESTTYPE statement to the job definition. Specify EMAIL or WEB.
2. Add the DISTRLISTROLES statement to distribute the report using a distribution list of roles.

   To specify multiple roles to distribute the report to, separate each role with a comma and enclose the entire list of roles in single quotes.
3. Add the DISTRLISTUSERS statement to distribute the report using a distribution list of operator IDs.

   To specify multiple operator IDs to distribute the report to, separate each operator ID with a comma and enclose the entire list of operator IDs in single quotes.
4. Add the EMAILADDR statement to distribute the report using a distribution list of email addresses.

   To specify multiple email addresses, separate each address with a semi-colon and enclose the entire list of addresses in single quotes.
5. (Optional) Add the following statements to specify additional distribution information:
   - DISTRFOLDER
   - EMAILLOG
   - EMAILSUBJECT
   - EMAILTEXT
   - EMAILWEBREPORT

Example: Distribute a Report to Operators

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.

```
AGENT PSAGENT
PROCESSTYPE Crystal
PROCESSNAME XRFWIN
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP3
RUNCONTROLID test
DISTRLISTUSERS 'VP1,VP2,VP3'
```

Example: Email a PeopleSoft Report

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 EMAILADDR statement. The email includes a subject title and body text.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

Send the Output of a PeopleSoft Job to a File

You can define a PeopleSoft job to run a process and send the output to a file.

To send the output of a PeopleSoft job to a file

1. Add the OUTDESTTYPE statement to the job definition. Specify FILE.
2. (Optional) Add the OUTDESTPATH statement to override the default PeopleSoft log/output directory. Specify the path to the output directory and the output file name.

Example: Send a Job's Output to a Specified File

This example runs an SQR Report. The report is formatted as PDF and outputted to a file.

```
AGENT PS_NY
PROCESSNAME XRFWIN
PROCESSTYPE 'SQR Report'
OUTDESTTYPE FILE
OUTDESTFORMAT PDF
OUTDESTPATH '/export/home/PSoutput/report1.pdf'
RUNCONTROLID test
PSOPRID VP1
```

Example: Send a Job's Output to the Default PeopleSoft Log/Output Directory

The example sends the job's output to the default PeopleSoft log/output directory because the job definition does not include the OUTDESTPATH statement:

```
AGENT PS_NY
OUTDESTTYPE FILE
OUTDESTFORMAT TXT
PROCESSNAME PAYROLL
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

Send the Output of a PeopleSoft Job to a Printer

You can define a PeopleSoft job to run a process and send the output to a printer.

To send the output of a PeopleSoft job to a printer

1. Add the OUTDESTTYPE statement to the job definition. Specify PRINTER.
2. (Optional) Add the OUTDESTPATH statement to override the default PeopleSoft printer. Specify the network location of the printer including the printer server and shared printer name.

   Note: The specified printer must be set up on the PeopleSoft system.

Example: Send a Job's Output to a Specified Printer

This example runs an SQR Report. The report is formatted as PS and outputted to a printer.

```
AGENT PS_NY
PROCESSNAME XRFWIN
PROCESSTYPE 'SQR Report'
OUTDESTTYPE PRINTER
OUTDESTFORMAT PS
OUTDESTPATH '\\CA\PRINTER1'
RUNCONTROLID test
PSOPRID VP1
```

Example: Send a Job's Output to the Default PeopleSoft Printer

The example sends the job's output to the default PeopleSoft printer because the job definition does not include the OUTDESTPATH statement:

```
AGENT PSAGENT
PROCESSNAME PAYROLL
PROCESSTYPE 'SQR Report'
SERVERNAME PSPR
OUTDESTTYPE PRINTER
OUTDESTFORMAT PS
RUNCONTROLID printtest
PSOPRID VP1
```

Chapter 11: SAP Jobs

This section contains the following topics:

SAP Jobs (see page 153)
Defining SAP Batch Input Session Jobs (see page 155)
Defining SAP Business Warehouse InfoPackage Jobs (see page 158)
Defining SAP Business Warehouse Process Chain Jobs (see page 159)
Defining SAP Data Archiving Jobs (see page 160)
Defining SAP Event Monitor Jobs (see page 162)
Defining SAP Job Copy Jobs (see page 164)
Defining SAP Process Monitor Jobs (see page 165)
Defining SAP R/3 Jobs (see page 166)
Define Recipients for the Job's Output (see page 169)
Email the Spool File on Step Completion or Failure (see page 170)
Using Success and Failure Messages within an SAP Job (see page 172)

SAP Jobs

SAP jobs let you run SAP workload.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

You can define the following SAP jobs:

SAP Batch Input Session

Imports data from external systems to the SAP system.

SAP Business Warehouse (BW) InfoPackage

Transfers data from a data source to an SAP Business Warehouse system.

SAP Business Warehouse (BW) Process Chain

Creates Process Chains on the SAP system.

SAP Data Archiving

Stores information in an SAP Archiving Object.

SAP Event Monitor

Monitors and triggers SAP events.

SAP Job Copy

Copies an existing SAP R/3 job.

SAP Process Monitor

Monitors for a specific SAP process status.

SAP R/3

Schedules an SAP R/3 job on your SAP system.

Defining SAP Batch Input Session Jobs

You can define an SAP Batch Input Session (BDC_JOB) job to import large amounts of data from external systems to the SAP system.

To schedule an SAP Batch Input Session job, you first define an ABAP that creates a Batch Input Session (BDC ABAP) on the SAP system. Next, you schedule an SAP Batch Input Session job in your scheduling manager to run the BDC ABAP on the SAP system. After the job runs, the Batch Input Session starts the data transfer.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Required Statements

To define an SAP Batch Input Session job, you must specify the following statements:

- AGENT (see page 226)
- ABAPNAME (see page 224)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP Batch Input Session job:

- ARCCLIENT (see page 229)
- ARCCONNECT (see page 230)
- ARCDATE (see page 231)
- ARCDOCCLASS (see page 232)
- ARCDOCTYPE (see page 233)
- ARCFORMAT (see page 234)
- ARCHOSTLINK (see page 235)
- ARCINFO (see page 237)
- ARCMODE (see page 238)
- ARCOBJTYPE (see page 240)
- ARCPATH (see page 242)
- ARCPRINTER (see page 243)
- ARCPROTOCOL (see page 244)
- ARCREPORT (see page 245)
- ARCSERVICE (see page 246)
- ARCSTORAGE (see page 247)

- ARCTEXT (see page 248)
- ARCUSER (see page 249)
- ARCVERSION (see page 250)
- BANNER (see page 268)
- BANNERPAGE (see page 269)
- BDCERRATE (see page 270)
- BDCEXTLOG (see page 271)
- BDCPROCRATE (see page 272)
- BDCSYSTEM (see page 273)
- COLUMNS (see page 288)
- EXPIRATION (see page 357)
- LANGUAGE (see page 402)
- LINES (see page 406)
- PRINTCOPIES (see page 462)
- PRINTCOVER (see page 463)
- PRINTDATASET (see page 464)
- PRINTDEPARTMENT (see page 465)
- PRINTDEST (see page 466)
- PRINTFOOTER (see page 468)
- PRINTFORMAT (see page 469)
- PRINTHOSTPAGE (see page 470)
- PRINTIMMED (see page 471)
- PRINTNEWSPOOL (see page 472)
- PRINTPRIORITY (see page 473)
- PRINTPW (see page 474)
- PRINTREL (see page 475)
- PRINTREQTYPE (see page 476)
- PRINTSPOOLNAME (see page 477)
- PUTINOUTBOX (see page 518)
- RECIPIENT (see page 521)
- RECIPIENTBCC (see page 522)
- RECIPIENTCC (see page 523)
- RECIPIENTEXPRESS (see page 524)

- RECIPIENTFORWARD (see page 525)
- RECIPIENTPRINT (see page 526)
- RECIPIENTTYPE (see page 527)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPEMAILADDR (see page 553)
- SAPFAILUREMSG (see page 554)
- SAPJOBCLASS (see page 555)
- SAPLANGUAGE (see page 557)
- SAPSUCCESSMSG (see page 558)
- SAPTARGETSYSTEM (see page 559)
- SAPUSER (see page 560)
- SPOOLRECIPIENT (see page 590)
- STARTMODE (see page 592)
- STEPUSER (see page 596)
- VARIANT (see page 641)

Example: Run an ABAP on the SAP System

This example defines an SAP Batch Input Session job that runs the BDCABAP ABAP. The APAP creates a Batch Input Session on the SAP system. After the job runs, the Batch Input Session starts the data transfer.

```
SAPJOBNAME BDCTEST
AGENT SAPAGENT
ABAPNAME BDCABAP
```

Defining SAP Business Warehouse InfoPackage Jobs

You can define an SAP Business Warehouse InfoPackage (BWIP_JOB) 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 Statements

To define an SAP Business Warehouse InfoPackage job, you must specify the following statements:

- AGENT (see page 226)
- INFOPACK (see page 381)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP Business Warehouse InfoPackage job:

- PARAM (see page 446)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPLANGUAGE (see page 557)
- SAPUSER (see page 560)

Example: Run an InfoPackage on the SAP System

This example runs the ZPAK_6C7ZW2JAXS90AK71WD1CYIN InfoPackage on the SAP system.

```
SAPJOBNAME BWIPTEST
AGENT SAPAGENT
INFOPACK 'ZPAK_6C7ZW2JAXS90AK71WD1CYIN'
```

Defining SAP Business Warehouse Process Chain Jobs

You can define an SAP Business Warehouse Process Chain (BWPC_JOB) job to run a sequence of background processes on the SAP system. Some SAP processes trigger events that can start other processes. An SAP Business Warehouse Process Chain 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 Statements

To define an SAP Business Warehouse Process Chain job, you must specify the following statements:

- AGENT (see page 226)
- CHAIN (see page 277)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP Business Warehouse Process Chain job:

- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPLANGUAGE (see page 557)
- SAPUSER (see page 560)

Example: Run a Process Chain on the SAP System

This example runs the PC_7A3929SH Process Chain on the SAP system.

```
SAPJOBNAME BWPCTEST
AGENT SAPAGENT
CHAIN 'PC_7A3929SH'
```

Defining SAP Data Archiving Jobs

You can define an SAP Data Archiving (SAPA_JOB) 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 Statements

To define an SAP Data Archiving job, you must specify the following statements:

- AGENT (see page 226)
- ARCMODE (see page 238)
- ARCOBJNAME (see page 239)
- ARCOBJVARIANT (see page 241)
- PRINTDEST (see page 466)
- SAPJOBNAME (see page 556)

Note: If you set the archiving mode to 'ARCHIVE' or 'BOTH' in the ARCMODE statement, the ARCDOCTYPE, ARCINFO, and ARCOBJTYPE statements are also required.

Optional Statements

You can specify the following optional statements for an SAP Data Archiving job:

- ARCCLIENT (see page 229)
- ARCCONNECT (see page 230)
- ARCDATE (see page 231)
- ARCDOCCLASS (see page 232)
- ARCDOCTYPE (see page 233)
- ARCFORMAT (see page 234)
- ARCHOSTLINK (see page 235)
- ARCINFO (see page 237)
- ARCOBJTYPE (see page 240)
- ARCPATH (see page 242)
- ARCPRINTER (see page 243)
- ARCPROTOCOL (see page 244)
- ARCREPORT (see page 245)
- ARCSERVICE (see page 246)

- ARCSTORAGE (see page 247)
- ARCTEXT (see page 248)
- ARCUSER (see page 249)
- ARCVERSION (see page 250)
- BANNER (see page 268)
- BANNERPAGE (see page 269)
- CHILDMONITOR (see page 280)
- COLUMNS (see page 288)
- EXPIRATION (see page 357)
- LINES (see page 406)
- PRINTCOPIES (see page 462)
- PRINTCOVER (see page 463)
- PRINTDATASET (see page 464)
- PRINTDEPARTMENT (see page 465)
- PRINTFOOTER (see page 468)
- PRINTFORMAT (see page 469)
- PRINTHOSTPAGE (see page 470)
- PRINTIMMED (see page 471)
- PRINTNEWSPOOL (see page 472)
- PRINTPRIORITY (see page 473)
- PRINTPW (see page 474)
- PRINTREL (see page 475)
- PRINTREQTYPE (see page 476)
- PRINTSPOOLNAME (see page 477)
- RECIPIENT (see page 521)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPLANUGAGE (see page 557)
- SAPTARGETSYSTEM (see page 559)
- SAPUSER (see page 560)

Example: Define an SAP Data Archiving Job

This example defines an SAP Data Archiving job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
SAPJOBNAME DATEST
AGENT SAPAGENT
ARCMODE PRINT
PRINTDEST LP01
ARCOBJNAME 'BC_ARCHIVE'
ARCOBJVARIANT 'BC_ARCVARIANT'
```

Defining SAP Event Monitor Jobs

You can define an SAP Event Monitor (SAPE_JOB) 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 Statements

To define an SAP Event Monitor job, you must specify the following statements:

- AGENT (see page 226)
- EVENT (see page 345)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP Event Monitor job:

- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPLANGUAGE (see page 557)
- SAPUSER (see page 560)

Example: Monitor an SAP Event

This example defines an SAP Event Monitor job that monitors the SAP event named SAPEvent. The job completes when the event is triggered (raised) on the SAP system.

```
SAPJOBNAME EMTEST
AGENT SAPAGENT
EVENT 'SAPEvent' REGISTER
```

Example: Trigger an SAP Event

This example triggers the SAP event named SAPEvent:

```
SAPJOBNAME EMTEST
AGENT SAPAGENT
EVENT 'SAPEvent' TRIGGER
```

Defining SAP Job Copy Jobs

You can define an SAP Job Copy (SAP_JOB) job to copy 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.

Required Statements

To define an SAP Job Copy job, you must specify the following statements:

- AGENT (see page 226)
- JOBCOPY (see page 394)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP Job Copy job:

- CHILDMONITOR (see page 280)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPFAILUREMSG (see page 554)
- SAPLANGUAGE (see page 557)
- SAPSUCCESSMSG (see page 558)
- SAPTARGETSYSTEM (see page 559)
- SAPUSER (see page 560)
- STARTMODE (see page 592)
- WEBPOSTING (see page 646)

Example: Define an SAP Job Copy Job

This example defines an SAP Job Copy job that copies the SAPTEST2 job with job count 00458131.

```
AGENT SAPHTAGENT
JOBCOPY JOBCOUNT(00458131)
SAPJOBNAME SAPTEST2
```

Defining SAP Process Monitor Jobs

You can define an SAP Process Monitor (SAPM_JOB) job to monitor for a specific SAP process status. The SAP Process Monitor job can monitor continuously or end after the first detection of a process. You can 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 Statements

To define an SAP Process Monitor job, you must specify the following statements:

- AGENT (see page 226)
- PROCESSMONITOR (see page 486)

Optional Statements

You can specify the following optional statements for an SAP Process Monitor job:

- ABAPNAME (see page 224)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPLANGUAGE (see page 557)
- SAPTARGETSYSTEM (see page 559)
- SAPUSER (see page 560)

Example: Monitor for SAP Process Status

This example defines an SAP Process Monitor job that monitors the RUNNING status. The job completes when the RUNNING status is detected.

```
AGENT SAPAGENT
PROCESSMONITOR STATUS(RUNNING)
```

Defining SAP R/3 Jobs

You can define an SAP R/3 (SAP_JOB) 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 Statements

To define an SAP R/3 job, you must specify the following statements:

- ABAPNAME (see page 224)

  Note: 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.
- AGENT (see page 226)
- SAPJOBNAME (see page 556)

Optional Statements

You can specify the following optional statements for an SAP R/3 job:

- ARCCLIENT (see page 229)
- ARCCONNECT (see page 230)
- ARCDATE (see page 231)
- ARCDOCCLASS (see page 232)
- ARCDOCTYPE (see page 233)
- ARCFORMAT (see page 234)
- ARCHOSTLINK (see page 235)
- ARCINFO (see page 237)
- ARCMODE (see page 238)
- ARCOBJTYPE (see page 240)
- ARCPATH (see page 242)
- ARCPRINTER (see page 243)
- ARCPROTOCOL (see page 244)
- ARCREPORT (see page 245)
- ARCSERVICE (see page 246)
- ARCSTORAGE (see page 247)
- ARCTEXT (see page 248)

- ARCUSER (see page 249)
- ARCVERSION (see page 250)
- BANNER (see page 268)
- BANNERPAGE (see page 269)
- CHILDMONITOR (see page 280)
- COLUMNS (see page 288)
- EMAILADDR (see page 330)
- EXPIRATION (see page 357)
- FAILUREMSG (see page 358)
- LANGUAGE (see page 402)
- LINES (see page 406)
- PRINTCOPIES (see page 462)
- PRINTCOVER (see page 463)
- PRINTDATASET (see page 464)
- PRINTDEPARTMENT (see page 465)
- PRINTDEST (see page 466)
- PRINTFOOTER (see page 468)
- PRINTFORMAT (see page 469)
- PRINTHOSTPAGE (see page 470)
- PRINTIMMED (see page 471)
- PRINTNEWSPOOL (see page 472)
- PRINTPRIORITY (see page 473)
- PRINTPW (see page 474)
- PRINTREL (see page 475)
- PRINTREQTYPE (see page 476)
- PRINTSPOOLNAME (see page 477)
- PUTINOUTBOX (see page 518)
- RECIPIENT (see page 521)
- RECIPIENTBCC (see page 522)
- RECIPIENTCC (see page 523)
- RECIPIENTEXPRESS (see page 524)
- RECIPIENTFORWARD (see page 525)
- RECIPIENTPRINT (see page 526)

- RECIPIENTTYPE (see page 527)
- RFCDEST (see page 546)
- SAPCLIENT (see page 552)
- SAPEMAILADDR (see page 553)
- SAPFAILUREMSG (see page 554)
- SAPJOBCLASS (see page 555)
- SAPLANGUAGE (see page 557)
- SAPSUCCESSMSG (see page 558)
- SAPTARGETSYSTEM (see page 559)
- SAPUSER (see page 560)
- SPOOLRECIPIENT (see page 590)
- STARTMODE (see page 592)
- STEPUSER (see page 596)
- SUCCESSMSG (see page 598)
- VARIANT (see page 641)
- WEBPOSTING (see page 646)

Example: Run an ABAP on the SAP System

This example runs the BTCTEST ABAP on the SAP system.

```
SAPJOBNAME SAPTEST
AGENT SAPAGENT
ABAPNAME BTCTEST
```

Define Recipients for the Job's Output

In an SAP R/3 or an SAP Batch Input Session job, you can define recipients for the job's output (spool file). The recipient can be an email address, an SAPoffice distribution list, or an SAPoffice user.

To define recipients for the job's output

1. Specify the following statements in the job definition:
   - RECIPIENTTYPE

     The recipient type can be SU (SAP user), DLI (SAPoffice distribution list), or INT (email address).
   - SPOOLRECIPIENT

     You can use multiple SPOOLRECIPIENT statements to specify multiple recipients. Alternatively, you can define a distribution list in SAP and use the DLI recipient type.
2. (Optional) Specify optional statements to control delivery options:
   - PUTINOUTBOX
   - RECIPIENTBCC
   - RECIPIENTCC
   - RECIPIENTEXPRESS (SU and DLI recipient types only)
   - RECIPIENTFORWARD (SU and DLI recipient types only)
   - RECIPIENTPRINT (SU and DLI recipient types only)

Example: Send the Spool File to a Distribution List

This example sends the spool file to the users on the payroll distribution list:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE DLI
SPOOLRECIPIENT payroll
ABAPNAME BTCTEST
```

Example: Send a Carbon Copy of the Spool File to an Email Address

This example sends a carbon copy of the spool file to the address jsmith@company1.com:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE INT
RECIPIENTCC Y
SPOOLRECIPIENT jsmith@company1.com
ABAPNAME BTCTEST
```

Example: Send the Spool File to an SAP User Using Express Delivery

This example sends the spool file to the SAP user J01Prod using express delivery. If J01Prod is logged into the SAP system, J01PROD will receive a message that the spool file has been sent.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
RECIPIENTEXPRESS Y
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

Email the Spool File on Step Completion or Failure

You can email the SAP spool file for a single step in an SAP R/3 job to recipients. A copy of the spool file is emailed on step completion or failure.

To email the SAP spool file on step completion or failure

1. Specify the ABAPNAME statement to start the step definition within the job definition.
2. Specify the EMAILADDR statement after the ABAPNAME statement.

   To specify multiple email addresses, separate each address with a comma and enclose the entire list of addresses in single quotation marks.
3. (Optional) Specify the following optional statements to determine whether the step is a success or failure based on a text string in the job's output:
   - SUCCESSMSG
   - FAILUREMSG

Example: Email the Spool File to Two Addresses on Step Completion or Failure

This example emails the spool output to the user1@example.com and user2@example.com addresses when the BTCTEST ABAP completes or fails:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   EMAILADDR 'user1@example.com,user2@example.com'
```

Using Success and Failure Messages within an SAP Job

You can check an SAP job's output for specific text strings to determine whether the job is a success or a failure. You specify the text string in the SAPSUCCESSMSG or SAPFAILUREMSG statements in the job definition. For example, suppose when you cancel an SAP job you want it to complete to release its successor job. You can specify the text string 'Job canceled' as a success message in the job definition. When you cancel the job, the agent checks the job's spool file, finds a match for 'Job canceled' and marks the job as complete.

You can also check the output of a step (ABAP) to determine whether the step is a success or failure. You specify the text string in the SUCCESSMSG or FAILUREMSG statements after an ABAPNAME statement in the job definition.

For more flexibility, you can specify regular expressions instead of simple text strings within the success message and failure message statements. For example, you can use a regular expression to search for multiple strings at the same time. 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.

Note: To enable regular expression processing, the agent administrator must configure the agent for the following parameter: sap.useRegularExpressions=true.

Examples: Using Regular Expressions

- This expression checks for "TEST" in the job output file. The first .* indicates that any number of characters can precede TEST. The second .* indicates that any number of characters can follow it.

  ```
  .*TEST.*
  ```

- This expression checks whether "not found" or "started" appears in the job output file.

  ```
  .*(not found|started).*
  ```

- This expression checks whether "Job canceled" appears in the job output file.

  ```
  .*Job\scanceled.*
  ```

Chapter 12: Secure File Transfers

This section contains the following topics:

Secure Copy and Secure FTP Jobs (see page 173)
Defining Secure Copy Jobs (see page 174)
Defining Secure FTP Jobs (see page 175)

Secure Copy and Secure FTP Jobs

You can define a job to securely transfer binary files between an agent computer and a remote computer. You can upload to or download data from a remote server. The data is encrypted during the transfer.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

You can define the following job types to securely transfer a file:

Secure Copy

Lets you securely transfer binary files using the Secure Copy Protocol (SCP). The SCP protocol does not support wildcard transfers.

Note: Your agent administrator must configure the agent as an FTP client using the Secure Copy Protocol.

Secure FTP

Lets you securely transfer binary files using the Secure File Transfer Protocol (SFTP). The SFTP protocol supports wildcard transfers, so you can upload multiple files to a remote computer or download multiple files to the agent computer.

Note: Your agent administrator must configure the agent as an FTP client using the Secure File Transfer Protocol.

Defining Secure Copy Jobs

You can define a Secure Copy (SCP_JOB) job to transfer binary files using the Secure Copy Protocol (SCP). The SCP protocol does not support wildcard transfers.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Statements

To define a Secure Copy job, you must specify the following statements:

- AGENT (see page 226)
- LOCALNAME (see page 410)
- REMOTEDIR (see page 529)
- REMOTENAME (see page 534)
- SERVERADDR (see page 572)
- TRANSFERDIRECTION (see page 625)
- USER

Optional Statements

You can specify the following optional statements for a Secure Copy job:

- JOBCLASS
- LOCALUSER (see page 412)
- SERVERPORT (see page 575)
- TARGETOSTYPE (see page 602)

Example: Download a File Using the Secure Copy Protocol

This example downloads the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP):

```
AGENT WINAGENT
LOCALNAME C:\temp\install.log1
REMOTENAME install.log
REMOTEDIR /root
SERVERADDR chi-linux
SERVERPORT 22
TRANSFERDIRECTION DOWNLOAD
USER causer
```

Defining Secure FTP Jobs

You can define a Secure FTP (SFTP_JOB) job to transfer binary files using the Secure File Transfer Protocol (SFTP). The SFTP protocol supports wildcard transfers, so you can upload multiple files to a remote computer or download multiple files to the agent computer.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

Required Statements

To define a Secure FTP job, you must specify the following statements:

- AGENT (see page 226)
- LOCALNAME (see page 410)
- REMOTEDIR (see page 529)
- SERVERADDR (see page 572)
- TRANSFERDIRECTION (see page 625)
- USER

Optional Statements

You can specify the following optional statements for a Secure FTP job:

- JOBCLASS
- LOCALUSER (see page 412)
- REMOTENAME (see page 534)

  Note: The REMOTENAME statement is required if the LOCALNAME statement does not contain a wildcard.
- SERVERPORT (see page 575)
- TARGETOSTYPE (see page 602)

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).

```
AGENT WINAGENT
TRANSFERDIRECTION UPLOAD
SERVERADDR hpsupport
REMOTEDIR /u/tmp
REMOTENAME logs.tar
LOCALNAME D:\temp\logs.tar
USER causer
```

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). Since the LOCALNAME statement contains a wildcard, the REMOTENAME statement is not specified.

```
AGENT WINAGENT
SERVERADDR aixunix
TRANSFERDIRECTION UPLOAD
USER causer
REMOTEDIR /u1/build/uploaded
LOCALNAME c:\temp\upload\*
```

Chapter 13: SNMP Jobs

This section contains the following topics:

SNMP Jobs (see page 177)
Defining SNMP Subscribe Jobs (see page 178)
Defining SNMP Trap Send Jobs (see page 179)
Defining SNMP Value Get Jobs (see page 181)
Defining SNMP Value Set Jobs (see page 183)

SNMP Jobs

The agent supports a built-in SNMP manager capability. You can enable the agent to act as an SNMP manager to emit and listen for SNMP traps in addition to its other roles. The agent supports SNMP v1, v2, and v3. After the agent is configured, you can define and run SNMP job types on the agent.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows.

You can define the following types of SNMP jobs:

SNMP Subscribe

Waits for an event to occur on a port. Using filters, you can trigger your policies according to the device that raised the event or the enterprise, generic, or specific identifiers of the SNMP trap. You can use the SNMP Subscribe job to monitor a network device for critical errors and automatically create a trouble ticket and perform level 1 diagnostics on the device.

SNMP Trap Send

Raises an SNMP event that can be detected by a network systems manager application. You can use the SNMP Trap Send job to create events for policies that need to be tracked using an SNMP monitoring product.

SNMP Value Get

Queries a network device for the value of a variable that is assigned to a Management Information Base (MIB) address. You can use the SNMP Value Get job to retrieve information about a network device to determine whether an administrator needs to be notified.

SNMP Value Set

Modifies a variable on a network device. The variable is assigned to the MIB address that you specify. You can use the SNMP Value Set job to update a variable that reports on the failure or success of a mission-critical policy.

Defining SNMP Subscribe Jobs

You can define an SNMP Subscribe (SNPC_JOB) job to subscribe for SNMP trap information.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows. Your agent administrator must configure the agent as an SNMP manager and configure the SNMP trap listener.

Note: For information on configuring the agent's trap listener for SNMP Subscribe jobs, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

Required Statements

To define an SNMP Subscribe job, you must specify the following statements:

- AGENT (see page 226)
- FILTER (see page 377)
- MIB (see page 421)

Optional Statements

You can specify the following optional statement for an SNMP Subscribe job:

- DESTINATION (see page 314)
- JOBCLASS

Example: Subscribe to the First SNMP Trap the Agent Receives

Suppose that you want the agent to subscribe to the first SNMP trap it receives. In this example, the name of the MIB file is RFC1213-MIB.mib. When the agent receives the first trap, the job completes.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
FILTER .*
```

Example: Subscribe to All SNMP Traps the Agent Receives

Suppose that you want the agent to subscribe to all SNMP traps it receives. In this example, the name of the MIB file is RFC1213-MIB.mib. Whenever the agent receives a trap, an alert named ALRT is sent.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib' CONTINUOUS(ALRT)
FILTER .*
```

Defining SNMP Trap Send Jobs

You can define an SNMP Trap Send (SNPE_JOB) job to send SNMP trap information.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows. Your agent administrator must configure the agent as an SNMP manager.

Required Statements

To define an SNMP Trap Send job, you must specify the following statements:

- AGENT (see page 226)
- MIB (see page 421)
- SNMPNODE (see page 586)

In addition, if your SNMP version is v1 or v2, you must also specify the following statement:

- COMMUNITY (see page 291)

If your SNMP version is v3, you must also specify the following statements:

- AUTHPROTOCOL (see page 267)
- PRIVPROTOCOL (see page 479)
- SNMPUSER (see page 589)

Optional Statements

You can specify the following optional statements for an SNMP Trap Send job:

- DESTINATION (see page 314)
- JOBCLASS
- PARAMETER (see page 453)

In addition, if your SNMP version is v3, you can also specify the following statement:

- ENGINEID (see page 337)

Example: Send an SNMP Trap to a Network Device

Suppose that you want to send the cybtrapstart trap to a network device using SNMP v3. In this example, five string parameters are passed to the trap. The host name of the network device is localhost and its port is 162. The job uses the AES privacy protocol and the SHA authentication protocol. The name of the MIB file is RFC1213-MIB.mib and the default engine ID is used. The credentials of user1 are used for authorization.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE cybtrapstart HOST(localhost) PORT(162) VERSION(3)
SNMPUSER user1
AUTHPROTOCOL SHA
PRIVPROTOCOL AES
ENGINEID
PARAMETER TYPE(snmp:string) VALUE(p1)
PARAMETER TYPE(snmp:string) VALUE(p2)
PARAMETER TYPE(snmp:string) VALUE(p3)
PARAMETER TYPE(snmp:string) VALUE(p4)
PARAMETER TYPE(snmp:string) VALUE(p5)
```

Defining SNMP Value Get Jobs

You can define an SNMP Value Get (SNPG_JOB) job to retrieve the value of an SNMP variable.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows. Your agent administrator must configure the agent as an SNMP manager.

Required Statements

To define an SNMP Value Get job, you must specify the following statements:

- AGENT (see page 226)
- SNMPNODE (see page 586)

In addition, if your SNMP version is v1 or v2, you must also specify the following statement:

- COMMUNITY (see page 291)

If your SNMP version is v3, you must also specify the following statements:

- AUTHPROTOCOL (see page 267)
- PRIVPROTOCOL (see page 479)
- SNMPUSER (see page 589)

Optional Statements

You can specify the following optional statements for an SNMP Value Get job:

- DESTINATION (see page 314)
- JOBCLASS
- MIB (see page 421)

In addition, if your SNMP version is v3, you can also specify the following statements:

- CONTEXTENGINEID (see page 297)
- CONTEXTNAME (see page 298)

Example: Query a Network Device for the Value of an SNMP Variable

Suppose that you want to know the value of the agentVersion variable hosted by a network device. In this example, the host name of the network device is host.example.com and its port is 161. The SNMP version is v2 and the read community is public. The name of the MIB file is RFC1213-MIB.mib, which is located on the agent computer.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE agentVersion HOST(host.example.com) PORT(161) VERSION(2)
COMMUNITY public
```

Example: Query a Network Device for the Values of a Whole SNMP Subtree

Suppose that you want to retrieve the values of a whole SNMP subtree starting with the pluginManagerPluginsTable OID.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE pluginManagerPluginsTable HOST(host.example.com) PORT(161) VERSION(2) +
  SUBTREE
COMMUNITY public
```

Defining SNMP Value Set Jobs

You can define an SNMP Value Set (SNPS_JOB) job to set the value of an SNMP variable.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows. Your agent administrator must configure the agent as an SNMP manager.

Required Statements

To define an SNMP Value Set job, you must specify the following statements:

- AGENT (see page 226)
- PARAMETER (see page 453)
- SNMPNODE (see page 586)

In addition, if your SNMP version is v1 or v2, you must also specify the following statement:

- COMMUNITY (see page 291)

If your SNMP version is v3, you must also specify the following statements:

- AUTHPROTOCOL (see page 267)
- PRIVPROTOCOL (see page 479)
- SNMPUSER (see page 589)

Optional Statements

You can specify the following optional statements for an SNMP Value Set job:

- DESTINATION (see page 314)
- JOBCLASS
- MIB (see page 421)

Note: The MIB statement is required if the SNMPNODE statement specifies an OID (object identifier) in string (non-numeric) format.

In addition, if your SNMP version is v3, you can also specify the following statements:

- CONTEXTENGINEID (see page 297)
- CONTEXTNAME (see page 298)

Example: Change the Value of an SNMP Variable using SNMP v2

Suppose that you want to set the value of the agentLogLevel variable to its highest level to diagnose a problem. In this example, the host name of the network device is host.example.com and its port is 161. The SNMP version is v2 and the read community is public. The name of the MIB file is RFC1213-MIB.mib, which is located on the agent computer.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE agentLogLevel HOST(host.example.com) PORT(161) VERSION(2)
COMMUNITY public
PARAMETER TYPE(snmp:int) VALUE(8)
```

Example: Change the Value of an SNMP Variable using SNMP v3

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

Chapter 14: UNIX Jobs

This section contains the following topics:

UNIX Jobs (see page 185)
Defining UNIX Jobs (see page 190)
Verify File Space Before a UNIX Job Starts (see page 191)
Pass Positional Arguments in a UNIX Job (see page 191)
Pass Environment Variables in a UNIX Job (see page 192)
Define Alternative Input and Output Sources (see page 194)
Send a User-Defined Exit Code (see page 195)
Specifying the Command or Script Name Without the Full Path in a UNIX Job (see page 196)
Specifying the Command or Script Name Using an Environment Variable (see page 197)
Set a UNIX Job's Process Priority (see page 197)
Running a Script Under a Specific User's Account (see page 198)
Modify Resource Limits (see page 199)
How to Customize the Job's Runtime Environment (see page 200)
Run a Perl Script on UNIX (see page 202)

UNIX Jobs

UNIX jobs let you run workload on UNIX computers. The job can run a script or execute a command.

Note: To run these jobs, your system requires CA WA Agent for UNIX or Linux.

When you define a UNIX job, you can specify settings including the following:

Positional Parameters

Defines variables to pass to a program at the time the program is invoked.

Environment Variables

Specifies variables that define the local environment where the job runs.

User-defined Exit Codes

Defines exit codes to indicate job success and job failure. By default, an exit code of 0 (zero) indicates job success and any other code indicates job failure.

Process Priority

Specifies the order in which processes are scheduled on the processor.

Shell

Specifies the shell the script runs in.

The Directory the Job Runs Under

The directory that a UNIX job runs under is determined by the following settings:

oscomponent.initialworkingdirectory Agent Parameter

Specifies the default initial working directory for all scripts. Options are the following:

- SCRIPT—Sets the path to where the script resides.
- USER—Sets the path to the home directory of the owner of the script.
- PATH—Sets the path to an absolute path to where the script runs.

If you do not specify a value, the parameter defaults to the path where the running cybAgent resides.

You can specify the oscomponent.initialworkingdirectory parameter in the agent's agentparm.txt file.

HOME and PWD Environment Variables

Overrides the oscomponent.initialworkingdirectory default directory. The directory depends on the following settings:

- If you specify a value for PWD (Present Working Directory) in the job definition, the job runs under the PWD you specified.
- If you specify a value for HOME in the job definition, but you do not specify a value for PWD, the job runs under the HOME directory.
- If you specify values for HOME and PWD in the job definition, the job runs under the PWD directory.

Determining Which Shell is Used

A shell is a program that provides an environment for users to interact with the system. Shell programs are files that contain a series of user commands that are grouped, using logic provided by the shell, to accomplish a task.

Shells can be defined by you, the script's programmer, your agent administrator, and your UNIX administrator. When you define a job that runs on UNIX, you may want to know which shell is used to run the script because different shells have different facilities and characteristics. Some functions are specific to certain shells and may be incompatible with other shells.

The shell used is determined by the following settings in the following order:

1. The SHELL statement (if specified in the job definition)
2. The first line of the script (if the SHELL statement is not specified)
3. The oscomponent.defaultshell parameter in the agent's agentparm.txt file (if not specified in the SHELL statement or in the script)
4. The user default shell defined in the user profile (if not specified in one of the previous three locations)

Notes:

- If the shell is defined in the first line of the script, you do not need to include the SHELL statement in the job definition.
- 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*.

Example: Specify the Shell in the SHELL Statement

In this example, the C shell is specified in the SHELL statement.

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test1.csh
SHELL /bin/csh
```

Example: Specify the Shell in the First Line of the Script

The first line of the following test1.csh script identifies the C shell:

```
#!/bin/csh -f
if ( $LOGNAME != guest) then
    echo "User is not guest"
endif
echo $LOGNAME logon
exit 0
```

The following job definition does not contain the SHELL statement:

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test1.csh
USER guest
```

In this example, agent UNIX_LA runs the test1.csh script using the C shell.

Shell Initialization Files

When you log in to the UNIX operating system, the operating system reads a series of initialization files depending on the shell you use.

C Shell Initialization Files

When you log in to the C shell, it automatically runs a number of files. The first file run is a system file named /etc/.login, which contains system-wide configuration information (such as your default path). After these files are run, the C shell reads and runs the commands from the following files in your home directory:

.cshrc

Establishes variables and operands that are local to a specific shell. Each time you create a new shell, the C shell reinitializes these variables for the new shell.

The following is a sample .cshrc file:

```
set noclobber
set ignoreeof
set path = (~/bin $path /usr/games)
alias h history
```

.login

Contains commands that you want to run once at the beginning of each session. If the C shell is running as a login shell, it reads and runs the commands in the .login file in your home directory after login.

The following is a sample .login file:

```
setenv history 20
setenv MAIL /usr/spool/mail/$user
```

.logout

Runs when you exit from your login shell.

Korn Shell Initialization Files

The Korn shell supports three startup scripts:

/etc/profile

Contains system-wide startup commands. This file is a login script and runs when you log in.

$HOME/.profile

Runs when you log in. Use this login script to set options, set the path, and set and export variable values.

The following is a sample $HOME/.profile file:

```
set -o allexport
PATH=.:/bin:/usr/bin:$HOME/bin
CDPATH=.:$HOME:$HOME/games
PS1='! $PWD> '
ENV=$HOME/.kshrc
TIMEOUT=0
set +o allexport
```

The script named in the Korn shell variable ENV

Runs when you create a Korn shell or run a Korn shell script. Use this file to define aliases and functions and to set default options and variables that you want to apply to the Korn shell invocation.

Bourne shell initialization files

When you log in to your system, the Bourne shell looks for the /etc/profile file. This file contains system-wide startup commands for the Bourne shell. Only the system administrator can change this file.

After running the commands in the /etc/profile file, the Bourne shell runs the commands in the $HOME/.profile file. Therefore, you can override the system-wide commands in the /etc/profile file with commands in the $HOME/.profile file.

Defining UNIX Jobs

You can define a UNIX (UNIX_JOB) job to schedule workload to run on a UNIX computer. The job can run a script or execute a command.

Note: To run these jobs, your system requires CA WA Agent for UNIX or Linux.

Required Statements

To define a UNIX job, you must specify the following statements:

- AGENT (see page 226)
- CMDNAME (see page 284) or SCRIPTNAME (see page 563)

Note: Use the SCRIPTNAME statement to run shell scripts and the CMDNAME statement to run binary files. If you specify the CMDNAME statement, you must also specify the USER statement.

Optional Statements

You can specify the following optional statements for a UNIX job:

- ARGS (see page 257)
- ENVAR
- EXITCODE
- FILESYSTEM (see page 374)
- JOBCLASS
- PROCESS_PRIORITY (see page 484)
- SHELL (see page 582)
- ULIMIT (see page 631)
- USER (required if CMDNAME is specified)

Example: Run a Command

This example runs the data command under the user1 user ID. The job runs on the UNIX_NY agent.

```
AGENT UNIX_NY
CMDNAME /mfg/test/data
USER user1
```

Note: When running a command, you must define the user ID the command runs under.

Example: Run a Script

This example invokes the C shell to run the sort script on the UNIX_LA agent:

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test/sort
SHELL /bin/csh
```

Verify File Space Before a UNIX Job Starts

You can define a UNIX job to check if one or more UNIX file systems have the required amount of available space. At run time, the agent checks whether the required space is available on the agent computer. If all of the requirements are met, the job starts. If any of the requirements are not met, the job fails.

By default, UNIX jobs do not check the available file space.

To verify file space before a UNIX job starts, specify the FILESYSTEM statement in the job definition. To check the available space on multiple file systems, define a separate FILESYSTEM statement for each file system.

Example: Verify Available File Space to Start a Job 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.

```
AGENT UNIXAGENT
SCRIPTNAME /u1/procrun.sh
FILESYSTEM /roots SIZE(100)
FILESYSTEM /auxfsl SIZE(120)
```

Pass Positional Arguments in a UNIX Job

When running workload, you might need to pass data between jobs and across platforms. You can pass positional arguments to a command or script in your job definition. Positional arguments are variables that can be passed to a program at the time the program is invoked. The arguments are assigned in the order they are passed.

To pass positional arguments to a command or script, specify the ARGS statement in the job definition. You can specify multiple ARGS statements in a UNIX job definition.

Example: Pass Positional Arguments to a UNIX Command

This example passes three arguments to a UNIX command. The argument "user 1" is enclosed with double quotation marks because it contains a space.

```
ARGS "user 1" 905-555-1212 749
```

When the command runs, the arguments are set as follows:

Argument	Value Passed
1	user 1
2	905-555-1212
3	749

Pass Environment Variables in a UNIX Job

In UNIX, you specify environment variables to define the local environment the script runs in. You can modify existing environment variables or create your own.

To pass an environment variable to a script, specify the ENVAR statement in the job definition. You can specify multiple ENVAR statements in a UNIX job definition.

Example: Pass UNIX Environment Variables to a Script

This example includes two ENVAR statements that pass environment variables to a script and a third ENVAR statement that defines the Present Working Directory (PWD). The parameter "user 1" is enclosed with double quotation marks because it contains a space.

```
SCRIPTNAME /home/scripts/pay
AGENT UNIX_NY
ENVAR NAME="user 1"
ENVAR JOB=PAYROLL
ENVAR 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

UNIX Environment Variables

When a UNIX job runs under a specific user account, the agent can pass the user's environment variables to the script or program. You can also set up a script's running environment by overriding the environment variables in the job definition. For example, you can override the HOME environment variable to run the script under a user's login directory.

You can pass the following UNIX environment variables in a job definition to override the variable values:

HOME

Identifies the user's login directory. You can override the HOME value to set up a user-specific environment by specifying a different login directory in the job definition.

Example: HOME=/home/guest/bin

Notes:

- You can set up the script's running environment in the .profile and .login files.
- You must also set the oscomponent.loginshell parameter to true in the agent's agentparm.txt file to run the login scripts located in the HOME directory.

PATH

Provides a list of directories that the shell searches when it needs to find a command. Each directory is separated by a colon. and the shell searches the directories in the order listed. The PATH variable is the most important environment variable. You can override the PATH value to set up a user-specific environment by specifying a different PATH in the job definition.

Note: Overriding the default system path can result in the "command not found" error.

ENV

Contains the name of the initialization file to run when a new Korn shell starts. You can override the ENV value to set up a user-specific environment by specifying a different ENV value in the job definition.

Example: ENV=/home/guest/bin/myenv

Note: The name of the file used to set up the script-running environment must be .profile. The .profile must be the same file used with the HOME variable.

PWD

Contains the absolute path name of your current directory.

Define Alternative Input and Output Sources

All UNIX programs run by a shell are connected to the following input and output streams:

Stream	Default Source
Standard input stream	Keyboard
Standard output stream	Screen
Standard error output stream	Screen

To define alternative input and output sources, specify the following environment variables in your job definition:

- STDIN, to specify an alternative input stream, for example:

  ```
  ENVAR STDIN=/path/name
  ```

- STDOUT, to specify an alternative output stream, for example:

  ```
  ENVAR STDOUT=/path/name
  ```

- STDERR, to specify an alternative error output stream, for example:

  ```
  ENVAR STDERR=/path/name
  ```

In the preceding examples, name is the file to be used as the standard input or output stream and path is the location of the file.

The STDOUT file name displays when you list the workload object with the LISTAPPL or APPLJOB command. The STDOUT file name may be different from what was specified in STDOUT=. For example:

```
ENVAR STDOUT=$AGENTPATH/stdout.txt
```

In this example, the environment variable AGENTPATH equals '/root/home/2'; therefore, the file name resolves to StdOut=/root/home/2/stdout.txt in the output of the workload object.

This is applicable only for Windows NT and all UNIX agents.

The STDERR file name displays when you list the workload object with the LISTAPPL or APPLJOB command. The STDERR file name may be different from what was specified in STDERR=. For example:

```
ENVAR STDERR=$AGENTPATH/stderr.txt
```

In this example, the environment variable AGENTPATH equals '/root/home/2'; therefore, the file name resolves to StdErr=/root/home/2/stderr.txt in the output of the workload object.

This is applicable only for Windows NT and all UNIX agents.

Send a User-Defined Exit Code

By default, an exit code of zero (0) is interpreted as job success and any other number indicates job failure. However, you can map exit codes other than 0 as job success.

To send a user-defined exit code, specify the EXITCODE statement in the job definition.

Example: Send a User-Defined Exit Code

The following example shows the first and last lines of the payroll.sh script. The script returns the self-defined exit code 100 to the scheduling manager.

```
#!/usr/bin/sh
.
.
.
exit 100
```

The following example shows the definition of the job that runs the script. In this job, the EXITCODE statement defines exit code 100 as SUCCESS, indicating successful completion of the script.

```
AGENT SUN_NY
SCRIPTNAME  /home/esp/payroll.sh
EXITCODE 100 SUCCESS
EXITCODE 0   SUCCESS
```

Specifying the Command or Script Name Without the Full Path in a UNIX Job

When defining a job, the agent 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 met:

- The agent is running under the root account.
- The agent is configured to resolve environment variables.

  Note: To configure the agent to resolve environment variables, ask your agent administrator to refer to the information about the oscomponent.lookupcommand parameter in the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- The user ID you enter in the USER statement has the authority to run the job on the agent. The user default shell is used.
- The path to the script or command name is set in the PATH system environment variable for the specified user ID.

Example: Run a Script that is Located in a Path Set in the PATH System Environment Variable

In this example, an agent named UNIXAGENT runs a script named procscript.sh. The job runs under the user ID jsmith, which has the authority to run the script. The path to procscript.sh is set in the PATH system environment variable for jsmith on the agent computer and the agent is configured to search for paths to command and script files.

```
AGENT UNIXAGENT
SCRIPTNAME procscript.sh
USER jsmith
```

Specifying the Command or 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 met:

- The agent is running under the root account.
- The agent is configured to resolve environment variables.

  Note: To configure the agent to resolve environment variables, ask your agent administrator to refer to the information about the oscomponent.lookupcommand parameter in the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- The user ID you enter in the USER statement has the authority to run the job on the agent computer. The user default shell is used.
- The environment variable you use, such as $MY_PATH, is set in the specified user ID's profile file.

Example: Run a Script that is Located in a Path Set in a User Environment Variable

In this example, an agent named UNIXAGENT runs a script named myscript.sh. The job runs under the user ID jsmith, which has the authority to run the script. The path to myscript.sh is set in the user environment variable $MY_PATH, which is defined in the profile file for jsmith, and the agent is configured to search for paths to command and script files.

```
AGENT UNIXAGENT
SCRIPTNAME $MY_PATH/myscript.sh
USER jsmith
```

Set a UNIX Job's Process Priority

You can set a UNIX job’s process priority. 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: You can only set a UNIX job’s process priority to a level above normal if the job runs on a computer with the agent started by the root account. If the agent is not started by root and you set the process priority to a level above normal, the job runs with the normal process priority.

To set a UNIX job’s process priority, specify the priority level using the PROCESS_PRIORITY statement in the job definition.

Example: Set the Process Priority for a UNIX Job

This example sets the process priority for a UNIX job to ABOVE_NORMAL:

```
AGENT LINUXAG
SCRIPTNAME /payroll/sort
PROCESS_PRIORITY ABOVE_NORMAL
```

Running a Script Under a Specific User's Account

Using the agent, there are two ways to run a command or script under a specific user's account:

- The agent is started as that user.
- The agent is started as root and the user is specified in the job definition with the USER statement.

Note: When the agent is started as root and there is no USER statement in the job definition, by default the agent runs the script as the owner of the script. For more information about this default, see the definition of the parameter oscomponent.noswitchsuid in the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

If the shell is specified in the first line of the script and its path matches the path defined in the oscomponent.validshell parameter, you do not have to use the SHELL statement to define the shell that you want the agent to use. For example, you can run a script using the environment defined in a specific user's account.

Example: Run a Script Under a Specific User Account

In this example, if the first line of the script is #!/bin/ksh and the path /bin/ksh is defined using the oscomponent.validshell parameter, you can use the USER statement to run the script under a specific user account, as follows:

```
AGENT SUN30RD
SCRIPTNAME /home/guest/bin/cmd1.ksh
USER guest
```

In this example, the agent runs the cmd1.ksh Korn script using the environment defined in either the $HOME/.login file or the
$HOME/.profile file, depending on which login shell is defined for user guest:

- If csh is defined, the $HOME/.login file is used.
- If ksh is defined, the $HOME/.profile file is used.

The login shell for user guest does not have to be the Korn shell. For example, the agent can pick up the following environment variables for user guest:

```
HOME=/home/guest
LOGNAME=guest
USER=guest
SHELL=/usr/bin/csh
PWD=/home/guest
```

In this example, user guest has specified the login shell as the C shell. The agent, therefore, runs the cmd1.ksh script using the environment defined in the $HOME/.login file.

Modify Resource Limits

When you define a UNIX job, you can set the job to modify resource limits. For example, you can define a job that modifies the maximum core file size and CPU time on the UNIX computer.

To modify resource limits in a UNIX job, specify the ULIMIT statement in the job definition.

Example: Specify Multiple ulimit Values

This example runs the procrun.sh script on the UNIXAGT agent. The job modifies the following resource limits on the UNIX computer:

- The core file size limit is 100 KB (soft limit). The size can increase to 200 KB (hard limit).
- The stack size limit is 250 KB (soft limit). The size can increase to 300 KB (hard limit).
- The CPU time limit is 1000 seconds (soft limit). The time can increase to 4000 seconds (hard limit).
- The process virtual size limit is 3332 KB (soft limit). The size can increase to an unlimited value.

```
AGENT UNIXAGT
SCRIPTNAME /ul/procrun.sh
ULIMIT TYPE(C) SOFT(100) HARD(200)
ULIMIT TYPE(S) SOFT(250) HARD(300)
ULIMIT TYPE(T) SOFT(1000) HARD(4000)
ULIMIT TYPE(M) SOFT(3332) HARD(unlimited)
```

How to Customize the Job's Runtime Environment

When you run a shell script under a user's account, you can customize the job's runtime environment. The agent runs the job using the specified user's login environment and the runtime environment you specify.

To customize the job's runtime environment, do the following:

- Set the oscomponent.loginshell parameter to true in the agentparm.txt file.
- Override the variables HOME and ENV in the job definition.

Customize the Runtime Environment for a Korn Shell Script

If your job runs a Korn shell script, you can customize the runtime environment by specifying a specific environment for the script.

To customize the runtime environment for a Korn shell script, specify the following statements in the job definition:

```
ENVAR HOME=/directory_name
ENVAR ENV=/directory_name/myenv
```

Before running the Korn script, the agent runs the /*directory_name*/.profile file and the /*directory_name*/myenv file.

Customize the Runtime Environment for a Bourne Shell Script

If your job runs a Bourne shell script, you can customize the runtime environment by specifying a specific environment for the script.

To customize the runtime environment for a Bourne shell script, do the following:

- Define a specific environment for the Korn shell
- Have the agent run the Bourne shell script using the Korn shell

Note: The ENV variable only works for the Korn shell. The Korn shell is a superset of the Bourne shell. Anything that runs under the Bourne shell runs without modification under the Korn shell.

Example: Customize the Runtime Environment for a Bourne Shell Script

In this example, the agent runs the cmd1.sh Bourne shell script. Before running the script, the agent runs the following:

- The login file for user guest
- The /esp/myenv file

```
AGENT SUN30RD
SHELL /bin/ksh
SCRIPTNAME /home/bin/cmd1.sh
USER guest
ENVAR ENV=/esp/myenv
```

Customize the Runtime Environment for a C Shell Script

If your job runs a C shell script, you can customize the runtime environment by specifying a specific environment for the script.

To customize the runtime environment for a C shell script, do the following:

- Set the environment in a .login file
- Set the HOME variable to the file's location in the ENVAR statement

Example: Customize the Runtime Environment for a C Shell Script

In this example, the agent picks up the environment for user root and runs the /home/bin/.login file before running the cmd1.csh C shell script.

```
AGENT SUN30
SCRIPTNAME /home/guest/bin/cmd1.csh
ENVAR HOME=/home/bin/
```

In this example, the agent is running as root. (Note that the login shell for root is not the C shell.)

Run a Perl Script on UNIX

You can define a UNIX job to run a Perl script.

To run a Perl script on UNIX

1. Ask your agent administrator to add the path of the Perl interpreter to the oscomponent.validshell parameter in the agent's agentparm.txt file, as shown in the following example:

   ```
   oscomponent.validshell=/usr/bin/sh,/bin/csh,/bin/ksh,/usr/local/bin/perl,/usr/local/bin/bash
   ```

   Note: If the oscomponent.checkvalidshell parameter is set to false, you do not need to perform the first step.

2. Specify the name of the Perl script you want to run using the SCRIPTNAME statement, as shown in the following example:

   ```
   SCRIPTNAME /bin/cmd1.pl
   ```

3. Do *one* of the following:

   - Specify the path to the Perl interpreter in the first line of the script, as shown in the following example:

     ```
     Perl Script cmd1.pl
       #!/usr/local/bin/perl
       print " $0 @ARGV\n";
       while (( $var, $value) = each %ENV) {
         print "$var = $value\n";
       }
       $live=$ENV{pick};
       print " user variable pick = $live\n";
     ```

   - Specify the path to the Perl interpreter in the job definition using the SHELL statement, as shown in the following example:

     ```
     SHELL /usr/local/bin/perl
     ```

Example: Run a Perl Script

This example runs the Perl script named cmd1.pl. The path to the Perl interpreter, /usr/local/bin/perl, is specified in the SHELL statement.

```
AGENT SUN30RD
SCRIPTNAME /bin/cmd1.pl
SHELL /usr/local/bin/perl
ARGS Hello world
USER guest
```

Chapter 15: Wake on LAN Jobs

This section contains the following topics:

Wake on LAN Jobs (see page 205)
Defining Wake on LAN Jobs (see page 206)

Wake on LAN Jobs

You can save energy using the agent's Wake on LAN (WOL) feature to automate the startup and shutdown of your computers. WOL lets you define and schedule WOL jobs to send a signal to a server to turn it on. When the server is no longer needed, you can schedule a different command job to power it down.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows. Your agent administrator must configure the agent to support WOL. For more information about configuring the agent to support WOL, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

Wake on LAN (WOL) is a hardware and software solution that lets you wake up a computer remotely. The solution requires an ACPI-compliant computer and a special software program that sends a signal to the computer's network card to wake it up. The agent provides the AMD magic packet to broadcast the signal to a computer that has been soft-powered-down (ACPI D3-warm state).

Defining Wake on LAN Jobs

You can define a Wake on LAN (WOL_JOB) job to send a signal to a server to turn it on. The job can wake up a remote computer that has been soft-powered-down (ACPI D3-warm state).

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows.

Required Statements

To define a Wake on LAN job, you must specify the following statements:

- AGENT (see page 226)
- BROADCAST (see page 275)
- MAC (see page 416)

Optional Statements

You can specify the following optional statements for a Wake on LAN job:

- JOBCLASS
- TIMEOUT (see page 616)
- WAKEHOST (see page 643)
- WAKEPASSWORD (see page 644)
- WAKEPORTS (see page 645)

Example: Broadcast the WOL Signal and Complete Immediately

This example broadcasts the WOL signal to the server identified by the 172.16.0.0 IP address and the 00-11-43-73-38-DC MAC address. After the WOL signal is sent, the job completes immediately without verifying whether the signal worked.

```
AGENT AGENTNME
BROADCAST 172.16.0.0
MAC 00-11-43-73-38-DC
```

Example: Broadcast the WOL Signal and Ping a Port

This example broadcasts the WOL signal to the server identified by the 172.16.00 IP address and the 00-1E-4F-C1-0F-FE MAC address. After the WOL signal is sent, the agent pings port 7 of the host computer to ensure it is available. If port 7 is available, the job completes successfully; otherwise, it fails. The job uses the default timeout for the ping.

```
AGENT AGENTNME
WAKEHOST host
WAKEPORTS 7
BROADCAST 172.16.0.0
MAC 00-1E-4F-C1-0F-FE
```

Chapter 16: Web Service Jobs

This section contains the following topics:

Web Service Jobs (see page 209)
Defining Web Service Jobs (see page 211)

Web Service Jobs

The term web service describes a standardized method for exchanging data between applications and systems. Web services use XML to code and decode the data and Simple Object Access Protocol (SOAP) to transfer it.

Web Service Description Language (WSDL) is an XML-based language that describes a web service and how to access it. A WSDL document specifies the location of the service and the operations the service exposes.

Universal Description, Discovery and Integration (UDDI) is an XML-based registry for businesses to list their available web services on the Internet. You can use the UDDI to access the WSDL.

Web services provide access to applications written in Java and Microsoft©.NET. A web service lets you invoke operations such as currency conversion, stock exchange quotes, or product pricing. In an enterprise workload automation environment, a web service might be used to invoke a business process such as posting accounts payable to the General Ledger. Some scheduling manager functions are also available as web services.

You can define a Web Service job to call an operation within a web service. The job passes parameters to the operation. The parameters can be actual values or a serialized Java object passed by another job. When the job invokes the web service, the parameters are passed to the operation. The job's output is stored by default as a serialized Java object in the job's spool directory. You can also specify a destination file for the output.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Web Services.

The following diagram shows the functional relationship between the scheduling manager, CA WA Agent for Web Services, and a web service residing on a web server:

Note: If your company has a firewall and you must communicate through a proxy server to access a computer outside the firewall, agent configuration is required. For more information on configuring the agent for a proxy, see the *CA Workload Automation Agent for Web Services Implementation Guide*.

Defining Web Service Jobs

You can define a Web Service (WEB_SERV) 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 Statements

To define a Web Service job, you must specify the following statements:

- AGENT (see page 226)
- OPERATION (see page 434)
- TARGETNAMESPACE (see page 601)

Optional Statements

You can specify the following optional statements for a Web Service job:

- DESTINATION (see page 314)
- ENDPOINT_URL (see page 336)
- FILTER (see page 375)
- JOBCLASS
- PARAMETER (see page 456)
- PORTNAME (see page 459)
- RETURNCLASS (see page 542)
- RETURNNAMESPACE (see page 543)
- RETURNXML (see page 544)
- SERVICENAME (see page 577)
- USER
- WSDL_URL (see page 647)

Note: In a Web Service job, if you specify the WSDL_URL statement but not the ENDPOINT_URL statement, you must specify both the SERVICENAME and PORTNAME statements.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE 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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webservicex.net
SERVICENAME ValidateEmail
PORTNAME ValidateEmailSoap
OPERATION IsValidEmail
WSDL_URL http://www.webservicex.net/ValidateEmail.asmx?wsdl
PARAMETER TYPE(xsd:string) VALUE(john.smith@example.com)
RETURNCLASS java.lang.Boolean
RETURNXML boolean
RETURNNAMESPACE http://www.webservicex.net
FILTER true
```

Chapter 17: Windows Jobs

This section contains the following topics:

Windows Jobs (see page 213)
Defining Windows jobs (see page 214)
Verify File Space Before a Windows Job Starts (see page 215)
Pass Positional Arguments in a Windows Job (see page 215)
Pass Environment Variables in a Windows Job (see page 216)
Specifying the Command or Script Name Without the Full Path in a Windows Job (see page 217)
Run the Windows Command Interpreter (see page 218)
Use a Windows Job Object to Manage Job Processing Properties (see page 218)
Access Network Resources (see page 220)

Windows Jobs

Windows jobs let you run workload on Windows computers. The job runs a Windows command file.

Note: To run these jobs, your system requires CA WA Agent for Windows.

When you define a Windows job, you can specify settings including the following:

Positional Parameters

Defines variables to pass to a program at the time the program is invoked.

Environment Variables

Specifies variables that define the local environment where the job runs.

User-defined Exit Codes

Defines exit codes to indicate job success and job failure. By default, an exit code of 0 (zero) indicates job success and any other code indicates job failure.

Windows Job Object

Defines a Windows job object that manages processing properties for a group of Windows jobs.

Defining Windows jobs

You can define a Windows (NT_JOB) job to schedule workload to run on a Windows computer. The job runs a Windows command file.

Note: To run these jobs, your system requires CA WA Agent for Windows.

Required Statements

To define a Windows job, you must specify the following statements:

- AGENT (see page 226)
- CMDNAME (see page 284)

Optional Statements

You can specify the following optional statements for a Windows job:

- ARGS (see page 257)
- ENVAR
- EXITCODE
- FILESYSTEM (see page 374)
- INTERACTIVE (see page 384)
- JOBCLASS
- JOBOBJECT (see page 398)
- USER

Example: Run a Windows Command

This example runs the c:\payroll\test\data.bat command on the NT_NY agent. The job runs on a Windows computer.

```
AGENT NT_NY
CMDNAME c:\payroll\test\data.bat
```

Verify File Space Before a Windows Job Starts

You can define a Windows job to check if one or more Windows drives have the required amount of available space. At run time, the agent checks whether the required space is available on the agent computer. If all of the requirements are met, the job starts. If any of the requirements are not met, the job fails.

By default, Windows jobs do not check the available file space.

To verify file space before a Windows job starts, specify the FILESYSTEM statement in the job definition. To check the available space on multiple drives, define a separate FILESYSTEM statement for each drive.

Example: Verify Available File Space to Start a Job 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.

```
AGENT WINAGENT
CMDNAME C:\Programs\Payroll\pay.exe
FILESYSTEM C SIZE(100)
FILESYSTEM D SIZE(120)
```

Pass Positional Arguments in a Windows Job

When running workload, you might need to pass data between jobs and across platforms. You can pass positional arguments to a command in your job definition. Positional arguments are variables that can be passed to a program at the time the program is invoked. The arguments are assigned in the order they are passed.

To pass positional arguments to a command, specify the ARGS statement in the job definition. You can specify multiple ARGS statements within a single job definition.

Example: Pass Positional Arguments to a Windows Program

This example passes three arguments to a Windows program. The argument "C:\Pay Data\salary.dat" is enclosed with double quotation marks because it contains a space.

```
ARGS "C:\Pay Data\salary.dat" 341 749
```

When the program runs, the arguments are set as follows:

Argument	Value Passed
1	C:\Pay Data\salary.dat

Argument	Value Passed
2	341
3	749

Pass Environment Variables in a Windows Job

In Windows, you specify environment variables to define the local environment the command or batch file runs in. You can modify existing environment variables or create your own.

To pass an environment variable to a command, specify the ENVAR statement in the job definition. You can specify multiple ENVAR statements in a Windows job definition.

Example: Pass Windows Environment Variables to a Batch File

This example includes three ENVAR statements that pass environment variables to a Windows batch file and a fourth ENVAR statement that defines the home directory:

```
CMDNAME c:\payroll\command1
AGENT NT_NY
ENVAR A=B
ENVAR C='X Y'
ENVAR E=pay
ENVAR HOME=c:\export\u1
```

In this example, the command command1 can reference these variables:

Environment Variable	Value Passed
A	B
C	'X Y'
E	pay
HOME	c:\export\u1

Specifying the Command or Script Name Without the Full Path in a Windows Job

When defining a job, the agent 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 met:

- The agent is configured to search for paths to command or script files.

  Note: To configure the agent to search for paths, ask your agent administrator to refer to the information about the oscomponent.lookupcommand parameter in the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

- 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

Example: Specify a Script Name Without the Full Path

In this example, an agent named WINAGENT runs a script named procscript.bat. The path to procscript.bat is set in the system path on the agent computer and the agent is configured to search for paths to command and script files.

```
AGENT WINAGENT
CMDNAME procscript.bat
```

Run the Windows Command Interpreter

You can schedule a job to run a command using the Windows command interpreter (cmd.exe).

To run the Windows command interpreter

1. Specify the path to cmd.exe using the CMDNAME statement in the job definition.

   The path to cmd.exe depends on your Windows operating system version. For example, on Windows NT, the path is C:\WINNT\system32\cmd.exe.

2. Pass arguments to cmd.exe using the ARGS statement.

   You must enclose the arguments in double quotation marks and precede the argument with the /C switch.

You can do this to run any cmd.exe command.

Example: Copy a File to Another Location

This example uses cmd.exe to copy a file to another location.

```
AGENT NT30
CMDNAME C:\Windows\system32\cmd.exe
ARGS "/C copy C:\env.txt C:\test\env.txt"
```

Use a Windows Job Object to Manage Job Processing Properties

A Windows job object lets you group processes together and control their attributes as a single entity. You can use a Windows job object to manage processing properties (such as processor usage, memory usage, and process priority) for a group of Windows jobs.

You can create a new Windows job object and associate a job with it or you can associate a job with an existing job object. After all processes associated with a job object complete, the job object no longer exists.

To define a Windows job object, add the JOBOBJECT statement in the job definition.

Example: Create a New Windows Job Object and Assign a Windows Job to It

The following job definition creates a Windows job object named payjobsobject. Payjobsobject can use up to 40 MB of memory (41943040 bytes) and 1 hour of CPU time (3600000 milliseconds) for all processes it contains. Each process associated with payjobsobject has a higher than normal priority and can use up to 500 KB of memory (512000 bytes) and 3 minutes of CPU time (180000 milliseconds). Payjobsobject can have up to 10 simultaneously active processes.

```
AGENT WINAGENT
CMDNAME myscript.bat
JOBOBJECT payjobsobject CREATE JOBMEMORY(41943040) +
  PROCESSMEMORY(512000) JOBTIME(3600000) PROCESSTIME(180000) +
  PRIORITY(ABOVE_NORMAL) PROCESSLIMIT(10)
```

Access Network Resources

When your agent runs as a Windows service, you can schedule Windows workload that accesses network resources. For example, you can specify UNC names and share names in your job definition.

Usually, when running Windows programs 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 with the USER statement in your job definition. To use the USER statement, your agent must run as a Windows service under the local system account (the default configuration).

To access Windows network resources

1. Verify with your agent administrator that the agent is running as a Windows service under the local system account.
2. Ask your scheduling manager administrator to define a user ID and password on the scheduling manager that has access to the file on the remote Windows system.
3. Specify the file name using the CMDNAME statement in the job definition.

   Notes:

   - 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.
   - You can specify share names. A share name is an alias for the path the resource exists in.
   - You can specify the share names C$ and ADMIN$ if the agent service logs on to a remote Windows server as a user with administrative authority. The agent can then access remote resources that are not marked as shared.
4. Specify the user ID and the domain the user ID belongs to using the USER statement in the job definition.

Notes:

- Before accessing network resources with your agent, verify that you are complying with the terms of your agent license agreement. In most situations, you are permitted to access data on remote computers; however, scripts or executable files run by an agent should use the CPU and memory of the computer where the agent resides.

- Although not recommended, your agent administrator can run the agent as a Windows service under a local user account (the This Account option). When you run the service under a local user account, when the service starts, it runs using the security context of the specified user account. If the user account and password are valid, the service process has access to network resources.
- When you access a remote computer from an agent on Windows, the user ID defined in the USER statement or in the This Account option is a domain user. If the local and remote servers are standalone servers, you must have the same user IDs and passwords defined on both servers.
- Some Windows 2000 applications do not run correctly when they are run remotely from the Windows NT 4.0 operating system.
- For more information on configuring and running the agent as a Windows service, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

Example: Run a Command on a Remote Server

In this example, the path c:\WINNT\Profiles\Visitor\Desktop\ has the share name MyDesktop. The command notify.cmd is in that path on the CYBNT server. JDOE is a user ID in the CYBDOM domain and has access to the notify.cmd command. JDOE's password is defined on the scheduling manager.

```
AGENT NT20
CMDNAME \\CYBNT\MyDesktop\notify.cmd
USER CYBDOM\JDOE
```

Example: Run an Executable in Public Folder on a Remote Server

This example runs calc.exe on the CYBNT server. CYBUSER is a user ID in the CYBDOM domain. CYBUSER is defined on the scheduling manager and has access permission to the public folder.

```
AGENT NT30
CMDNAME \\CYBNT\public\calc.exe
USER CYBDOM\CYBUSER
```

Example: Access a Remote Resource Using the C$ Share Name

In this example, drive C is accessed by an administrator over the network through an agent. The agent is running under the System Account option. The agent runs the test application in the c:\working directory on the server CYBNT. The directory c:\working is not a shared resource. The user admin1 is a valid user on both the local and remote computers and belongs to the Administrators group. admin1 is also in the CYBDOM domain.

```
AGENT NT30
CMDNAME \\CYBNT\C$\working\test
USER CYBDOM\admin1
```

Specifying a Password for a User ID

You can define a Windows job to access Windows network resources by specifying the user ID and the domain name the user ID belongs to using the USER statement. The resources are accessed under the specified user ID.

Note: To use the USER statement in a Windows job, your agent must run as a Windows service under the local system account.

If the user ID requires a password, your administrator must define the password on the scheduling manager. For security reasons, you do not define the password in the job definition. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

When you specify a user ID that requires a password in a job definition, the scheduling manager sends the agent the user ID and password pair (the password is encrypted). The scheduling manager searches the repository for an entry matching the specified user name with the job type and a qualifier equal to the agent name. This search returns a password that most closely matches the search parameters. If there are no matching entries, an entry with no type or qualifier is returned.

The following job types require a user password:

- Database
- FTP
- JMX
- PeopleSoft
- SAP
- Windows

Note: If you define the user ID in the Services setup on the Windows computer instead of using the USER statement, you need to define the password in the Services setup also.

Chapter 18: C-LANG Statements

Syntax Diagrams

The syntax diagrams in this guide use the following conventions:

Notation	Meaning
Quote marks “ or '	Must be entered as shown.
Comma ,	Must be entered as shown.
Ellipsis …	The operand can be repeated. Do not enter ellipsis.
Lowercase italics	An operand must be substituted. User-supplied variable or character string.
Uppercase operand	The operand must be spelled as shown. You can enter the command and the operand in either upper or lower case.
OR-bar (|)	Indicates an exclusive value on left or right of bar. You must enter one of the items. You cannot enter more than one.
Underline _____	If you do not enter one of the operands, the system supplies the underlined operand; this is the default.
Parentheses ( ) and special characters	Operand enclosed in parentheses is mandatory and must be entered as shown.
Single operand in square brackets [ ]	Optional operand; do not type the brackets.
Stacked operands in braces { } { }	Mandatory; you must enter one of the operands. You cannot enter more than one.
Stacked operands in square brackets [ ] [ ]	Optional operand; you can enter one value, or none.
Operands with OR-bars (|) and square brackets [ ]|[ ]	Optional, mutually exclusive operands. Enter one or none.

Notation	Meaning
Stacked operands in square brackets within braces {[ ] [ ]}	Mandatory; you must enter one of these operands. You can enter more than one.

ABAPNAME Statement—Specify an ABAP Name

The ABAPNAME statement starts a step definition within the SAP job definition and identifies the ABAP step to be run.

Supported Job Types

This statement is required for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

This statement is optional for the SAP Process Monitor job type (see page 165).

Syntax

This statement has the following format:

```
ABAPNAME ABAPname
```

ABAPname

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 40 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Define multiple ABAPs within a job definition using multiple ABAPNAME statements.
- When defining an SAP R/3 job, 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.

Example: Run an ABAP Step

This example defines a job that runs the RSPARAM ABAP:

```
SAPJOBNAME CYBJG01.JOB1
AGENT HPSAP
ABAPNAME RSPARAM
PRINTDEST ATLY
```

Example: Run Multiple ABAP Steps

This example defines an SAP R/3 job that runs two ABAPs with different print parameters:

```
SAPJOBNAME CYBJG01.JOB2
AGENT HPSAP
ABAPNAME RSPARAM
   PRINTDEST ATLY
ABAPNAME RSPARAG
   PRINTCOPIES 2
```

ACTION Statement—Specify a Servlet Path

The ACTION statement specifies the path to the servlet to be invoked in an HTTP job.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
ACTION servlet_path
```

servlet_path

Specifies the path to the program or servlet to be invoked.

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you omit this statement from the job definition, you must specify the program or servlet path (action) in the SERVLET_URL statement. If the ACTION statement is omitted from the job definition, the agent assumes that the SERVLET_URL statement specifies the full path to the program or servlet.
- If you include this statement in the job definition, do not specify the program or servlet path (action) in the SERVLET_URL statement. If the ACTION statement is included in the job definition, the agent assumes that the SERVLET_URL statement specifies only the server host name.

Example: Specify the Servlet Path the HTTP Job Invokes

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.

```
AGENT APPAGENT
INVOCATIONTYPE POST
SERVLET_URL http://localhost:8080
ACTION /examples/servlets/servlet/TheServlet
PARAMETER KEYWORD(key1) VALUE(subscribe)
PARAMETER KEYWORD(key2) VALUE(test@example.com)
```

AGENT Statement—Specify the Agent Where the Job Runs

The AGENT statement identifies the distributed platform where the job runs under the control of the agent.

Supported Job Types

This statement is required for all agent job types.

Syntax

This statement has the following format:

```
AGENT agent_name
```

agent_name

Specifies a valid agent name as defined in the agent definition.

Limits: Up to 16 characters; the first character must be a letter. The remaining characters can be any combination of alphanumeric characters, including the underscore character. The name cannot contain spaces or special characters.

Example: Define Agent Workload

This example runs the i5/OS job under the AS401 agent:

```
AGENT AS401
CLPNAME MFGPROG
```

APPLDISPLNAME Statement—Specify the Display Name of an Oracle Applications Application

The APPLDISPLNAME statement specifies the display name of the Oracle Applications application.

Supported Job Types

If the APPLSHORTNAME statement is not used, this statement is required for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Note: All Oracle E-Business Suite Request Set or Single Request job definitions require an APPLDISPLNAME statement or an APPLSHORTNAME statement. Use the APPLDISPLNAME statement to specify the display name of the application and the APPLSHORTNAME statement to specify the short name of the application.

Syntax

This statement has the following format:

```
APPLDISPLNAME display_name
```

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 240 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- DISPLNAME is an alias of APPLDISPLNAME.

Example: Specify the Display Name of an Oracle Applications Application

In the following example, the display name of the Oracle Applications application is Application Object Library.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

APPLSHORTNAME Statement—Specify the Short Name of an Oracle Applications Application

The APPLSHORTNAME statement specifies the short name of the Oracle Applications application.

Supported Job Types

If the APPLDISPLNAME statement is not used, this statement is required for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Note: All Oracle E-Business Suite Request Set or Single Request job definitions require an APPLDISPLNAME statement or an APPLSHORTNAME statement. Use the APPLDISPLNAME statement to specify the display name of the application and the APPLSHORTNAME statement to specify the short name of the application.

Syntax

This statement has the following format:

```
APPLSHORTNAME short_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

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- SHORTNAME is an alias of APPLSHORTNAME.

Example: Specify the Short Name of an Oracle Applications Application

In the following example, the short name of the Oracle Applications application is ACCOUNTS.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLSHORTNAME ACCOUNTS
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

ARCCLIENT Statement—Specify the SAP Archive Link Client

The ARCCLIENT statement specifies the SAP archive link client.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCCLIENT client
```

client

Specifies the archive link client.

Limits: Up to three digits

Example: 800

Notes:

- Use the ARCCLIENT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCCLIENT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Client

This example specifies 800 as the archive link client for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCCONNECT Statement—Specify the SAP Archive Link Communication Connection

The ARCCONNECT statement specifies the SAP archive link communication connection.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCCONNECT connection
```

connection

Specifies the archive link communication connection.

Limits: Up to 14 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCCONNECT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCCONNECT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Communication Connection

This example specifies arcconnect as the archive link communication connection for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
   ARCCONNECT arcconnect
```

ARCDATE Statement—Specify the SAP Archive Link Archiving Date

The ARCDATE statement specifies the SAP archive link archiving date.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCDATE date
```

date

Specifies the archive link archiving date. The format is *yyyymmdd*.

Limits: Up to eight numeric digits

Example: 20080702

Notes:

- Use the ARCDATE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCDATE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Archiving Date

This example specifies April 8, 2004 as the archive link archiving date for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCDATE 20040408
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCDOCCLASS Statement—Specify the SAP Archive Link Document Class

The ARCDOCCLASS statement specifies the SAP archive link document class.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCDOCCLASS doc_class
```

doc_class

Specifies the archive link document class.

Limits: Up to 20 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCDOCCLASS statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCDOCCLASS statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Document Class

This example specifies arcdocclass as the archive link document class for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCDOCCLASS arcdocclass
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCDOCTYPE Statement—Specify the SAP Archive Document Type

The ARCDOCTYPE statement specifies the document type the SAP external system archive uses.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCDOCTYPE doc_type
```

doc_type

Specifies the name of the document type. The value corresponds to the Doc. type field on the Define Background Archive Parameters dialog.

Limits: Up to 10 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCDOCTYPE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCDOCTYPE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Document Type

This example specifies ARCHIVE as the archive document type for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   ARCDOCTYPE ARCHIVE
```

ARCFORMAT Statement—Specify the SAP Archive Output Format

The ARCFORMAT statement specifies the SAP archive output format.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCFORMAT output_format
```

output_format

Specifies the archive output format.

Limits: Up to 16 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCFORMAT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCFORMAT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Output Format

This example specifies arcformat as the archive output format for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCFORMAT arcformat
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCHOSTLINK Statement—Specify the SAP Archive Link RPC Host

The ARCHOSTLINK statement specifies the SAP archive link RPC host.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCHOSTLINK host
```

host

Specifies the archive link RPC host.

Limits: Up to 32 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCHOSTLINK statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCHOSTLINK statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link RPC Host

This example specifies rpchost as the archive link RPC host for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCHOSTLINK rpchost
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCINFO Statement—Specify the SAP Archive Link Information

The ARCINFO statement specifies the archive link information for the SAP external archive system.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCINFO arcinfo
```

arcinfo

Specifies the archive link information.

Limits: Up to 3 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCINFO statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCINFO statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Information

This example specifies INF as the archive link information for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   ARCINFO INF
```

ARCMODE Statement—Specify the SAP Archive Mode

The ARCMODE statement specifies the archive mode the SAP external archive system uses.

Supported Job Types

This statement is required for the SAP Data Archiving job type (see page 160).

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCMODE PRINT|ARCHIVE|BOTH
```

PRINT

Specifies that spool output is to be printed.

ARCHIVE

Specifies that spool output is to be archived.

BOTH

Specifies that spool output is to be printed and archived.

Notes:

- Use the ARCMODE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- Use the ARCMODE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- The archive mode corresponds to the SAPGUI Spool options, Archiving mode field on the Background Print Parameters dialog.

Example: Specify the Archive Mode

In this example, the spool output for the BTCTEST ABAP is printed.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   ARCMODE PRINT
```

ARCOBJNAME Statement—Identify Name of SAP Archiving Object

The ARCOBJNAME statement identifies the name of an archiving object defined on the SAP system.

Supported Job Type

This statement is required for the SAP Data Archiving job type (see page 160).

Syntax

This statement has the following format:

```
ARCOBJNAME 'object_name'
```

object_name

Specifies the name of the archiving object.

Limits: Up to 10 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- OBJNAME is an alias of ARCOBJNAME.

Example: Specify an Archiving Object

This example defines an SAP Data Archiving job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
SAPJOBNAME DATEST
AGENT SAPHTAGENT
ARCMODE BOTH
PRINTDEST LP01
ARCOBJNAME 'BC_ARCHIVE'
ARCOBJVARIANT 'BC_ARCVARIANT'
```

ARCOBJTYPE Statement—Specify the Type of SAP External System Archive Object

The ARCOBJTYPE statement specifies the type of SAP external system archive object.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCOBJTYPE object_type
```

object_type

Specifies the type of external system archive object. The value corresponds to the Obj. type field on the Define Background Archive Parameters dialog.

Limits: Up to 10 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCOBJTYPE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCOBJTYPE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Type of External System Archive Object

This example specifies ARCHIVE as the type of external system archive object for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   ARCOBJTYPE ARCHIVE
```

ARCOBJVARIANT Statement—Identify Name of SAP Archiving Object Variant

The ARCOBJVARIANT statement identifies the name of an archiving object variant defined on the SAP system.

Supported Job Type

This statement is required for the SAP Data Archiving job type (see page 160).

Syntax

This statement has the following format:

```
ARCOBJVARIANT 'object_variant'
```

object_variant

Specifies the name of the archive object variant.

Limits: Up to 14 valid SAP characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify an Archiving Object Variant

This example defines an SAP Data Archiving job that stores information described in the BC_ARCHIVE Archiving Object into an SAP data archive. The archiving object variant is BC_ARCVARIANT.

```
SAPJOBNAME DATEST
AGENT SAPHTAGENT
ARCMODE PRINT
PRINTDEST LP01
ARCOBJNAME 'BC_ARCHIVE'
ARCOBJVARIANT 'BC_ARCVARIANT'
```

ARCPATH Statement: Specify the SAP Standard Archive Path

The ARCPATH statement specifies the SAP standard archive path.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCPATH path
```

path

Specifies the standard archive path.

Limits: Up to 70 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCPATH statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCPATH statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Standard Archive Path

This example specifies /export/home/archive as the standard archive path for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCPATH /export/home/archive
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCPRINTER Statement—Specify the SAP Archive Target Printer

The ARCPRINTER statement specifies an SAP archive target printer.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCPRINTER printer
```

printer

Specifies the archive target printer.

Limits: Up to 4 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCPRINTER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCPRINTER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Target Printer

This example specifies PRT1 as the archive target printer for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCPRINTER PRT1
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCPROTOCOL Statement—Specify the SAP Archive Storage Connection Protocol

The ARCPROTOCOL statement specifies the SAP archive storage connection protocol.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCPROTOCOL protocol
```

protocol

Specifies the archive storage connection protocol.

Limits: Up to 8 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCPROTOCOL statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCPROTOCOL statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Storage Connection Protocol

This example specifies prtcl12 as the archive storage connection protocol for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCPROTOCOL prtcl12
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCREPORT Statement—Specify the SAP Archive Link Report Name

The ARCREPORT statement specifies the SAP archive link report name.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCREPORT report_name
```

report_name

Specifies the archive link report name.

Limits: Up to 40 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCREPORT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCREPORT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Report Name

This example specifies Archive Report as the archive link report name for the RSPARAM ABAP. The archive link report name requires single quotes because of the space.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCREPORT 'Archive Report'
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCSERVICE Statement—Specify the SAP Archive Link RPC Service/Destination

The ARCSERVICE statement specifies the SAP archive link RPC service/destination.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCSERVICE service
```

service

Specifies the archive link RPC service/destination.

Limits: Up to 32 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCSERVICE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCSERVICE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link RPC Service/Destination

This example specifies rpcdest as the archive link RPC service/destination for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCSERVICE rpcdest
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCSTORAGE Statement—Specify the SAP Archive Link Target Storage System

The ARCSTORAGE statement specifies the SAP archive link target storage system.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCSTORAGE storage_system
```

storage_system

Specifies the archive link target storage system.

Limits: Up to 2 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCSTORAGE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCSTORAGE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Target Storage System

This example specifies st as the archive link target storage system for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCSTORAGE st
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCTEXT Statement—Specify the SAP Archive Link Text Information Field

The ARCTEXT statement specifies the SAP archive link Text Information field.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCTEXT text_field
```

text_field

Specifies the archive link Text Information field.

Limits: Up to 40 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCTEXT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCTEXT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Link Text Information Field

This example specifies Archive text as the archive link Text Information field for the RSPARAM ABAP. The archive link Text Information field requires single quotes because of the space.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCTEXT 'Archive text'
   ARCMODE ARCHIVE
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO inf
   ARCCLIENT 800
```

ARCUSER Statement—Specify the SAP Archive Data Element for User

The ARCUSER statement specifies the archive data element for user.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

ARCUSER *user*

user

Specifies the archive data element for user.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCUSER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCUSER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Data Element for User

This example specifies arcuser as the archive data element for user for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCUSER arcuser
   ARCMODE BOTH
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO 123
```

ARCVERSION Statement—Specify the SAP Archive Version

The ARCVERSION statement specifies the archive version.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
ARCVERSION version
```

version

Specifies the archive version.

Limits: Up to 4 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the ARCVERSION statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the ARCVERSION statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Archive Version

This example specifies 4.1 as the archive version for the RSPARAM ABAP. The archive version requires single quotes because of the period.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   ARCVERSION '4.1'
   ARCMODE BOTH
   ARCDOCTYPE ARCHIVE
   ARCOBJTYPE ARCHIVE
   ARCINFO 123
```

ARGDEFAULTS Statement—Specify Whether to Use Oracle Applications Argument Defaults

The ARGDEFAULTS statement specifies whether to use default values for arguments not defined using the ARGS statement or the PROGARGS statement. The default arguments are defined in Oracle Applications.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
ARGDEFAULTS 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.

Notes:

- Your agent administrator can specify that all Oracle E-Business Suite Request Set or Single Request jobs use the default argument values by setting the oa.useArgDefaults parameter to true in the agentparm.txt file of the agent.
- The ARGDEFAULTS statement overrides the default setting specified in the oa.useArgDefaults parameter in the agentparm.txt file of the agent.
- The ARGDEFAULTS statement is ignored when Y or N is omitted.

Example: Use Oracle Applications Argument Defaults

In this example, each program has three arguments. ARGDEFAULTS is set to Y. Thus the agent passes default values from Oracle Applications to arguments not defined in the PROGARGS statement.

In the first program, the PROGARGS statement passes values for the first and third arguments. The agent passes a default value for the second argument.

In the second program, the PROGARGS statement passes values for the second argument. The agent passes default values for the first and third arguments.

```
AGENT CYBOA
REQUESTSET XXEXTRACTS
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
PROGDATA 1
   PROGARGS X45S,,X45G
PROGDATA 2
   PROGARGS ,X22F,
```

ARGS Statement—Specify a Parameter Passed to a Stored Procedure (Database Jobs)

The ARGS statement specifies a parameter passed to a database stored procedure in a Stored Procedure job.

Supported Job Type

This statement is optional for the Database Stored Procedure job type (see page 71).

Syntax

This statement has the following format:

```
ARGS [IGNORE] parm_name [IN|OUT|INOUT] data_type[,value]
```

IGNORE

(Optional) Specifies that the value is not returned.

Note: This operand only applies to output parameters.

parm_name

Specifies the name of the parameter.

IN|OUT|INOUT

(Optional) 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. Alternatively, you can specify IN OUT.

data_type

Specifies the database data type of the parameter.

The following data types are supported:

- CHAR
- VARCHAR
- LONGVARCHAR
- NUMERIC
- DECIMAL
- BIT
- BOOLEAN
- TINYINT

- SMALLINT
- INTEGER
- BIGINT
- REAL
- FLOAT
- DOUBLE
- DATE
- TIME
- TIMESTAMP

value

(Optional) Specifies the value of the parameter.

Limits: Up to 100 characters; case-sensitive

Note: This value applies to input parameters only.

Notes:

- To specify multiple parameters to pass to a stored procedure, define a separate ARGS statement for each parameter.
- The value of each statement can be up to 4070 characters.

Example: Pass Parameter to a Stored Procedure

This example defines a Stored Procedure job. The agent runs the stored procedure PAYROLL, passing in the parameter 'ename INOUT VARCHAR,John Evans'. Success is determined by whether ename begins with the string John when PAYROLL completes.

```
AGENT CYBDB1
SPNAME PAYROLL
ARGS ename INOUT VARCHAR,John Evans
JOB_CRITERIA ename=John.*
```

ARGS Statement—Define Argument Values to Pass (Oracle E-Business Suite Jobs)

The ARGS statement defines the argument values to override Oracle Applications default values.

Supported Job Type

This statement is optional for the Oracle E-Business Suite job type (see page 142).

Syntax

This statement has the following format:

```
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, separated by commas.
- Do not add a space after a comma, unless the argument value starts with a space.
- To use a default value defined in Oracle Applications, enter one comma in the argument position and specify Y in the ARGDEFAULTS statement. Do not add spaces between commas.
- The entire value can be up to 4078 characters. If your list of arguments is over the total maximum of 4078 characters, split it into multiple ARGS statements. These multiple statements will be concatenated using commas.
- You can specify up to 100 ARGS statements in the job definition.

Example: Pass Argument Values

In this example, the ARGS statement passes values for the first, third, and fifth arguments. Since the ARGDEFAULTS statement is set to Y, the agent passes default values to the second and fourth arguments.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
DESCRIPTION 'CONFIRMED REQ 20040808 by TYB212'
ARGS X22F,,X56R143,,X23T
ARGSDEFAULTS Y
```

ARGS Statement—Pass Additional Parameters for the PeopleSoft Report (PeopleSoft Jobs)

The ARGS statement specifies additional parameters to pass to a PeopleSoft report.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
ARGS argument ...
```

argument

Specifies additional argument values to pass to the PeopleSoft report.

Limits: Up to 4078 characters; case-sensitive

Note: You can pass multiple strings as a single argument by enclosing them in double quotes: “parm1 parm2”

Notes:

- The entire value can be up to 4078 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 double quotes.

- When you pass multiple arguments, you can use continuation characters such as the hyphen (-).
- If you specify more than one ARGS statement, the second ARGS statement within the job definition overrides the first, with the exception of a blank character string, which is ignored.

Example: Ignore Parameters in the PS_PRCSDEFN Table and Pass Additional Parameters to the nVision Report

In this example, the job parameters are not updated with data in the PS_PRCSDEFN table. The ARGS statement passes additional parameters to the nVision report.

```
AGENT PSAGENT
PROCESSTYPE nVision-Report
PROCESSNAME NVSRUN
OUTDESTTYPE FILE
OUTDESTFORMAT XLS
OUTDESTPATH c:\test\testnv.xls
SKIPPARMUPDATES Yes
ARGS -NRNVARIABLE -NBUAUS01 -NHLhttp://10.1.1.40/psp/ps+
    /EMPLOYEE/ERP/c/REPORT_BOOKS.IC_RUN_DRILLDOWN.GBL?Action=A
```

ARGS Statement—Pass Arguments (UNIX, Windows, and HP Integrity NonStop Jobs)

The ARGS statement specifies a string of positional arguments to pass to a batch file, script, command, or program.

Supported Job Types

This statement is optional for the following job types:

- UNIX (see page 190)
- Windows (see page 214)
- HP Integrity NonStop (see page 105)

Syntax

This statement has the following format:

```
ARGS argument ...
```

argument

Specifies the string argument to pass to the batch file, script, command, or program.

Limits: Case-sensitive

Note: You can pass multiple strings as a single argument by enclosing them in double quotes: “parm1 parm2”

Notes:

- The entire value can be up to 4078 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 double quotes.
- When you pass multiple arguments, you can use continuation characters such as the hyphen (-).
- If you specify more than one ARGS statement, the second ARGS statement within the job definition overrides the first, with the exception of a blank character string, which is ignored.
- For HP NonStop jobs, if you use an ARGS statement to directly invoke multiple commands, separate each command with a tilde and a semicolon (~;). All of these commands should be surrounded with one set of single quotes. For example:

  ```
  ARGS ‘#KEYS~;#MYTERM’
  ```

Example: Pass Positional Arguments to a Windows Program

This example passes three arguments to a Windows program. The argument "C:\Pay Data\salary.dat" is enclosed with double quotation marks because it contains a space.

```
ARGS "C:\Pay Data\salary.dat" 341 749
```

When the program runs, the arguments are set as follows:

Argument	Value Passed
1	C:\Pay Data\salary.dat
2	341
3	749

Example: Pass Positional Arguments to a UNIX Command

This example passes three arguments to a UNIX command. The argument "user 1" is enclosed with double quotation marks because it contains a space.

```
ARGS "user 1" 905-555-1212 749
```

When the command runs, the arguments are set as follows:

Argument	Value Passed
1	user 1
2	905-555-1212
3	749

Example: Pass Positional Arguments to CA WA Agent for HP Integrity NonStop

This example passes one argument to CA WA Agent for HP Integrity NonStop. The argument "PARM1 PARM2" is enclosed with double quotation marks because it contains a space.

```
ARGS "PARM1 PARM2"
```

AS400FILE Statement—Identify i5/OS Source File

The AS400FILE statement specifies the source file for the program that you want to run on an i5/OS system.

Supported Job Type

If the CLPNAME or COMMAND statement is not used, this statement is required for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
AS400FILE file|
          file(member)|
          library/file(member)|
          /QSYS.LIB/library.LIB/file.FILE/member.MBR
```

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.

library

Specifies the name of the library that contains the source file for the program that you want to run. The value must be a valid i5/OS library name.

Notes:

- The entire value can be up to 60 characters; it cannot contain delimiters (such as spaces).
- The AS400FILE statement is mutually exclusive with the CLPNAME and COMMAND statements.
- If the library name is already specified in the AS400FILE statement, do not use the AS400LIB statement. If you specify the library name in both statements, you will get a submission error.
- If you do not specify the library name in the AS400FILE statement or the AS400LIB statement, the agent uses the job's library list.

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 AS400FILE or the AS400LIB statement, so the agent uses the job's library list.

```
AGENT I5AGENT
AS400FILE CLSRC(BACKUP)
```

AS400LIB Statement—Specify the Name of a Library

The AS400LIB statement specifies the name of the library that contains the Control Language (CL) program, the source file for the program, or the command that you want to run.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
AS400LIB library|
        /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 AS400LIB statement, you can specify the library name in the CURLIB or LIBL statement.
- You can also specify the library name in the AS400FILE, CLPNAME, or COMMAND statement.
- Do not use the AS400LIB statement if the library name is already specified in the AS400FILE, CLPNAME, or COMMAND statement. If you specify the library name in both statements, you will get a submission error.
- If you do not specify the library name in the AS400FILE, CLPNAME, COMMAND, or AS400LIB statement, 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.

```
AGENT I5AGENT
CLPNAME MFGDATA
AS400LIB PRODJOBS
```

ASSIGN Statement—Assign a Logical File Name to a Physical File

The ASSIGN statement enables you to assign a logical file name to a physical file. Additionally, you can specify the characteristics of the physical file. The ASSIGN statement eliminates the requirement to hard-code file names in the program.

Supported Job Types

This statement is optional for the HP Integrity NonStop job (see page 105) type.

Syntax

This statement has the following format:

```
ASSIGN "logical-file-name;file-name
       [;file-attribute=value;file-attribute=value...]"
```

logical-file-name

Specifies the logical file name that is assigned to a physical file.

Limits: Maximum of 31 characters. Can contain alphanumeric characters, hypens (-), and circumflexes (^).

file-name

Specifies the name of the physical file.

Limits: The physical file name must have three qualifiers separated by periods. Each qualifier cannot be longer than eight characters.

file-attribute=value

Specifies an attribute of the physical file. The attributes are as follows:

PEXT

Specifies the size of the primary file extent that is allocated to the file.

Limits: An integer in the range from 1 through 65535

Example: PEXT=1024

SEXT

Specifies the size of the secondary file extent that is allocated to the file.

Limits: An integer in the range from 1 through 65535

Example: SEXT=512

EXL

Specifies the exclusion mode for the logical-unit, which determines the circumstances under which other processes can access the file.

Limits: Value can be EXCLUSIVE, SHARED, or PROTECTED

Example: EXL=EXCLUSIVE

ACC

Specifies the access mode for the logical-file-name, which specifies the file operations that can be performed.

Limits: Value can be I-O, INPUT, or OUTPUT

Example: ACC=I-O

CODE

Specifies the file code to assign to the logical-file-name.

Limits: Maximum is 22222

Example: CODE=101

REC

Specifies the record size.

Limits: An integer in the range from 1 through 65535

Example: REC=300

BLOCK

Specifies the size of data blocks that the logical unit uses.

Limits: Any integer in the range from 1 through 65535.

Example: BLOCK=4096

Notes:

- The text string within the double quotes of the ASSIGN statement cannot be longer than 1024 characters.
- You can specify up to 63 ASSIGN statements in one job.
- The ASSIGN statement is case-sensitive and it can contain the delimiters comma, semicolon, and space.

Example: ASSIGN Statement

In this example, the ASSIGN statement assigns logical name $VOL.SUBVOL.FILE1 to physical file $VOL1.SUBVOL1.FILE2.

```
AGENT PROAGENT
USER prod.glsys
COMMAND $C35.PROD.OABC
ENVAR STDOUT=$C35.PROD.cafout1
ENVAR STDERR=$c35.PROD.caferr1
ASSIGN "$VOL.SUBVOL.FILE1;$VOL1.SUBVOL1.FILE2"
PARAMETER NAME(ABC) VALUE(100)
DEFINE "MYFIL1;CLASS=MAP;FILE=$DATA2.PROD.FINFOJOB"
```

ATTRIBUTE Statement—Specify the MBean Attribute to Query or Set

The ATTRIBUTE statement specifies the name of the MBean attribute you want to query or set in a JMX-MBean Attribute job. You specify the ATTRIBUTE statement 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 statement is required for the following job types:

- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)

Syntax

This statement has the following format:

```
ATTRIBUTE attribute
```

attribute

Specifies the name of the MBean attribute that you want to query or set.

Limits: Up to 1024 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
ATTRIBUTE cachesize
```

ATTRIBUTESFILTER Statement—Specify the Attribute Notifications to Subscribe to

The ATTRIBUTESFILTER statement specifies a list of attribute notifications to subscribe to in a JMX-MBean Subscribe job. A JMX-MBean Subscribe job can monitor an MBean for a single notification or monitor continuously for notifications. You can use the ATTRIBUTESFILTER statement to filter the notifications by attribute. For example, you can use the ATTRIBUTESFILTER statement to monitor when attributes change.

Supported Job Type

This statement is optional for the JMX-MBean Subscribe job type (see page 58).

Syntax

This statement has the following format:

```
ATTRIBUTESFILTER attribute|'attribute,attribute...'
```

attribute

Identifies an attribute to monitor for change.

Limits: Case-sensitive

Notes:

- Enclose value that contain delimiters (such as spaces) in single quotation marks.
- In a list, enclose values that contain commas in single quotes.
- The entire value can be to 1024 characters.
- In a JMX-MBean Subscribe job, you can specify the ATTRIBUTESFILTER statement or the TYPESFILTER statement, but not both. If you do not specify either statement, the job will subscribe to all notifications for the MBean.

Example: Monitor for Changes to a Specific MBean Attribute Using a Filter

Suppose that you want to set up continuous monitoring for changes to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. Each time the cachesize attribute changes, an alert named CHGA is sent.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config' CONTINUOUS(CHGA)
ATTRIBUTESFILTER cachesize
```

AUTHORDER Statement—Specify Authentication Protocols

The AUTHORDER statement specifies a list of protocols to be used by a web server for authentication in an HTTP job.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following formats:

```
AUTHORDER BASIC|DIGEST|NTLM
```

```
AUTHORDER ({[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, place the protocols within parentheses, separated by commas or blanks.
- If you are connecting to a 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

AUTHPROTOCOL Statement—Specify the SNMP v3 Authentication Protocol

The AUTHPROTOCOL statement specifies the SNMP v3 authentication protocol to use when connecting with the user specified in the SNMPUSER statement. This statement is ignored if your SNMP version is v1 or v2.

Supported Job Types

If your SNMP version is v3, this statement is required for the following job types:

- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
AUTHPROTOCOL MD5|SHA
```

MD5

Specifies the Message Digest 5 (MD5) authentication protocol.

SHA

Specifies the Secure Hash (SHA) authentication protocol.

Example: Specify Authentication Protocol in an SNMP Value Set Job

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

BANNER Statement—Specify Whether to Include an SAP Cover Page

The BANNER statement specifies whether to include an SAP cover page. The SAP cover page contains information such as recipient name, department name, and format used.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
BANNER Y|N|D
```

Y

Prints the cover page.

N

Does not print the cover page.

D

Uses the default cover sheet output as determined by the setting of the selected output device.

Notes:

- Use the BANNER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the BANNER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- BANNER corresponds to the SAPGUI Cover sheets, SAP cover page field on the Background Print Parameters dialog.

Example: Print a Cover Page

This example prints the SAP cover page:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   BANNER Y
```

BANNERPAGE Statement—Specify Whether to Include an SAP Cover Page with the Report Output

The BANNERPAGE statement specifies whether to include an SAP cover page with the report output.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
BANNERPAGE Y|N
```

Y

Prints the cover page with the report output.

N

Does not print the cover page with the report output.

Notes:

- Use the BANNERPAGE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the BANNERPAGE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- BANNERPAGE corresponds to the SAPGUI Cover sheets, Selection cover page field on the Background Print Parameters dialog.

Example: Print a Cover Page With the Report Output

This example prints the cover page with the report output:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   BANNERPAGE Y
```

BDCERRRATE Statement—Specify the Maximum Acceptable Error Rate

The BDCERRRATE statement specifies the maximum acceptable error rate in an SAP Batch Input Session job.

Supported Job Type

This statement is optional for the SAP Batch Input Session job type (see page 155).

Syntax

This statement has the following format:

```
BDCERRRATE 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

Default: 0

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.

```
SAPJOBNAME BDCTEST
AGENT SAPHTAGENT
BDCERRRATE 5
ABAPNAME BDCABAP
```

BDCEXTLOG Statement—Generate Advanced Logging of the Batch Input Session

The BDCEXTLOG statement specifies whether to generate advanced logging of the Batch Input Session (BDC) running on the SAP system.

Supported Job Type

This statement is optional for the SAP Batch Input Session job type (see page 155).

Syntax

This statement has the following format:

```
BDCEXTLOG 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:

```
SAPJOBNAME BDCTEST
AGENT SAPHTAGENT
BDCEXTLOG Y
ABAPNAME BDCABAP
```

BDCPROCRATE Statement—Specify the Minimum Acceptable Process Rate

The BDCPROCRATE statement specifies the minimum acceptable process rate in an SAP Batch Input Session (BDC) job.

Supported Job Type

This statement is optional for the SAP Batch Input Session job type (see page 155).

Syntax

This statement has the following format:

```
BDCPROCRATE 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

Default: 0

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.

```
SAPJOBNAME BDCTEST
AGENT SAPHTAGENT
BDCPROCRATE 90
ABAPNAME BDCABAP
```

BDCSYSTEM Statement—Specify the Name of the Background Server

The BDCSYSTEM statement specifies the name of the background server that processes the Batch Input Session (BDC) running on the SAP system.

Supported Job Type

This statement is optional for the SAP Batch Input Session job type (see page 155).

Syntax

This statement has the following format:

```
BDCSYSTEM destination
```

destination

Specifies the name of the background server.

Limits: Up to 256 valid SAP characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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:

```
SAPJOBNAME BDCTEST
AGENT SAPHTAGENT
BDCSYSTEM dest
ABAPNAME BDCABAP
```

BEAN Statement—Specify the JNDI Name of the Bean

The BEAN statement specifies the JNDI name of the bean in an Entity Bean and Session Bean job.

Supported Job Types

This statement is required for the following job types:

- Entity Bean (see page 39)
- Session Bean (see page 65)

Syntax

This statement has the following format:

```
BEAN bean
```

bean

Specifies the JNDI name of the session or entity bean.

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- BEANNAME is an alias of BEAN.

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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(a23)
```

BEANNAME Statement—Specify the JNDI Name of the Bean

BEANNAME is an alias of BEAN (see page 274).

BROADCAST Statement—Specify the Broadcast Address

The BROADCAST statement specifies the IP address of the LAN or subnet of the computer that receives the Wake on LAN (signal).

Supported Job Type

This statement is required for the Wake on LAN job type (see page 206).

Syntax

This statement has the following format:

```
BROADCAST ip_address
```

ip_address

Specifies the IP address of the LAN or subnet of the computer that receives the Wake on LAN (WOL) signal. On UNIX, you can use ifconfig to obtain the broadcast address. On Windows, you must calculate the broadcast address by performing a bitwise OR operation on the IP address and the bit complement of the subnet mask. You can obtain the IP address and subnet mask using ipconfig.

Limits: Up to 100 characters

Example: Broadcast the WOL Signal to an IP Address

This example broadcasts the WOL signal to the server identified by the 172.16.0.0 IP address and the 00-11-43-73-38-DC MAC address. After the WOL signal is sent, the job completes immediately without verifying whether the signal worked.

```
AGENT AGENTNME
BROADCAST 172.16.0.0
MAC 00-11-43-73-38-DC
```

CCEXIT Statement—Specify Type of Exit Code Returned by an i5/OS Job

The CCEXIT statement 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 CCEXIT statement in your job definition, the job sends the ending severity code as the exit code.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
CCEXIT *PROGRAM|*SEVERITY|*USER
```

***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.

***SEVERITY**

Sends the job's ending severity code.

***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.

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, an exit code of zero (0) is interpreted 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

In this example, SALARY is a C language program. The I5AG agent sends the SALARY program's return code to the scheduling manager.

```
AGENT I5AG
CLPNAME SALARY
CCEXIT *USER
```

Example: Send an OPM COBOL Program's Return Code as the Job's Exit Code

In this example, PAYROLL is an OPM COBOL program. The I5AG agent sends the PAYROLL program's return code to the scheduling manager.

```
AGENT I5AG
CLPNAME PAYROLL
CCEXIT *PROGRAM
```

CHAIN Statement—Identify Name of Business Warehouse Process Chain on SAP System

The CHAIN statement identifies the name of a Business Warehouse Process Chain on the SAP system.

Supported Job Type

This statement is required for the SAP Business Warehouse Process Chain job type (see page 159).

Syntax

This statement has the following format:

```
CHAIN 'process_chain' [Simulate(Y|N)]
```

process_chain

Specifies the name of the Business Warehouse Process Chain.

Limits: Up to 25 valid SAP characters including any surrounding quotes; case-sensitive

Simulate(Y|N)

(Optional) Specifies whether to simulate or run the specified Process Chain on the SAP system.

- Y—Simulates the specified Process Chain on the SAP system.
- N—Runs the specified Process Chain on the SAP system.

Note: PROCESSCHAIN is an alias of CHAIN.

Example: Run a Process Chain on the SAP System

This example runs the PC_7A3929SH Process Chain on the SAP system:

```
SAPJOBNAME BWPCTEST
AGENT SAPHTAGENT
CHAIN 'PC_7A3929SH'
```

Example: Simulate a Process Chain on the SAP System

This example simulates the PC_7A3929SH Process Chain on the SAP system:

```
SAPJOBNAME BWPCTEST
AGENT SAPHTAGENT
CHAIN 'PC_7A3929SH' Simulate(Y)
```

CHILDMONITOR Statement—Specify Whether to Monitor the Children of Programs (Oracle E-Business Suite Jobs)

The CHILDMONITOR statement specifies whether the children of the Oracle Applications programs are monitored. Program children are programs that are released by the parent program.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Copy Job (see page 136)
- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
CHILDMONITOR 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.

Notes:

- Up to five levels of children can be monitored for each job.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.monChildren parameter in the agentparm.txt file of the agent.
- The CHILDMONITOR statement overrides the default setting specified in the oa.default.monChildren parameter in the agentparm.txt file of the agent.
- When monitoring children, include the MONITORDELAY statement in your job definition so the agent receives accurate information about the status of program children.
- When the CHILDMONITOR statement is defined in an Oracle E-Business Suite Request Set job, the setting is applied to all of the programs in the request set. You cannot specify a different setting for each program.
- The CHILDMONITOR statement is ignored when Y or N is omitted.

Example: Monitor the Children of a Program in an Oracle E-Business Suite Single Request Job

In this example, the children of the single request program are monitored. The agent checks the status of the program's children 60 seconds after the program completes.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
CHILDMONITOR Y
MONITORDELAY 60
```

Example: Monitor the Children of a Program in an Oracle E-Business Suite Copy Job

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

CHILDMONITOR Statement—Specify Whether to Monitor Children Jobs (SAP Jobs)

The CHILDMONITOR statement specifies whether to monitor children jobs. Children jobs are jobs spawned by a parent job.

Supported Job Types

This statement is optional for the following job types:

- SAP Data Archiving (see page 160)
- SAP Job Copy (see page 164)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
CHILDMONITOR Y|N
```

Y

Monitors the job's children.

N

Does not monitor the job's children.

Notes:

- Your agent administrator can specify a default setting for all applicable SAP jobs by setting the sap.job.children.monitor parameter in the agentparm.txt file of the agent.
- The CHILDMONITOR statement overrides the default setting specified in the sap.job.children.monitor parameter in the agentparm.txt file of the agent.
- The CHILDMONITOR statement is ignored when Y or N is omitted.

Example: Monitor Children Jobs

This example monitors the children spawned by the job:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
CHILDMONITOR Y
ABAPNAME ZCHILDJOB1
```

CLASSNAME Statement—Specify a Class Name

The CLASSNAME statement 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 statement is required for the following job types:

- JMX-MBean Create Instance (see page 55)
- POJO (see page 60)

Syntax

This statement has the following format:

```
CLASSNAME 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

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
CLASSNAME java.lang.Integer
METHOD parseInt
PARAMETER TYPE(java.lang.String) VALUE(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".

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
MBEAN 'DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic'
CLASSNAME cdc.jmx.SimpleDynamic
PARAMETER TYPE(java.lang.String) VALUE(Hello)
```

CLPNAME Statement—Identify i5/OS Program to Run

The CLPNAME statement specifies the program that you want to run on an i5/OS system.

Supported Job Type

If the AS400FILE or COMMAND statement is not used, this statement is required for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
CLPNAME program|
        library/program|
        /QSYS.LIB/library.LIB/program.PGM
```

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 that you want to run. The value must be a valid i5/OS library name.

Notes:

- The entire value can be up to 60 characters; it cannot contain delimiters (such as spaces).
- The CLPNAME statement is mutually exclusive with the AS400FILE and COMMAND statements.
- If you do not specify the library name in the CLPNAME statement or the AS400LIB statement, the agent uses the job's library list.
- If the library name is already specified in the CLPNAME statement, do not use the AS400LIB statement. If you specify the library name in both statements, you will get a submission error.

Example: Identify i5/OS 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.

```
AGENT I5AGENT
CLPNAME MFGDATA
AS400LIB PRODJOBS
```

CMDNAME Statement—Specify Commands to Run on Windows and UNIX

The CMDNAME statement specifies a command to run on a Windows or UNIX computer.

Supported Job Types

This statement is required for the Windows job type (see page 214).

To run binary files on UNIX, this statement is required for the UNIX job type (see page 190).

Note: All UNIX job definitions require a SCRIPTNAME statement or a CMDNAME statement. Use the SCRIPTNAME statement to run shell scripts and the CMDNAME statement to run binary files.

Syntax

This statement has the following format:

`CMDNAME` *`command_file`*

command_file

Specifies the full path and name of the command to run. It must be a legal executable Windows file name or a valid binary UNIX file name.

Limits: Up to 512 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Encryption must be enabled on the agent to use the CMDNAME statement.
- COMMAND and COMMANDNAME are aliases of CMDNAME.

UNIX Notes:

- When running a binary file using the CMDNAME statement, you must include the USER statement in the job definition. The USER statement specifies the user ID that the command runs under.
- The CMDNAME statement usually requires the full path to the command name you want to run. However, you can specify the command 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 USER statement, 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.

- 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.
- You can specify the command 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 USER statement, 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.

Windows Notes:

- You can run the following executable file types using the CMDNAME statement:
  - Binary files (.exe)
  - Batch files (.bat, .cmd)
  - Command files (.com)
- To run Windows applications, such as ipconfig.exe, specify the command file on the CMDNAME statement, as shown in the following example:

```
CMDNAME c:\winnt\system32\ipconfig.exe
```

- To run Windows operating system commands, such as DIR or TYPE, invoke the command interpreter CMD.EXE, as shown in the following example:

```
CMDNAME c:\winnt\system32\cmd.exe
ARGS /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 CMDNAME statement. (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 with the CMDNAME statement and use the ARGS statement to pass the command as a parameter.
- For some GUI applications, if there is a command line interface, you can run the application by specifying the batch file name using the CMDNAME statement. For example, you can transfer files using the FTP DOS commands or send e-mail in a batch file using software such as BatMail.

- 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\esp\espcmd1.bat
    C:\Program files\esp\espcmd2.bat
    C:\payroll\data\pay.exe
    ```

  - Use the Change Directory (cd) command to switch to the directory:

    ```
    cd C:\Program files\esp
    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.

- The CMDNAME statement 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
- Environment variables are not currently supported in the CMDNAME statement for Windows jobs.

Example: Identify a Command to Run Within a Job

In this example, sort is the name of the command and c:\payroll\test is the path.

```
CMDNAME c:\payroll\test\sort
```

Example: Run a Visual Basic Script

In this example, the agent invokes the Visual Basic interpreter wscript.exe using the CMDNAME statement and passes the Visual Basic script notify.vbs as a parameter using the ARGS statement:

```
AGENT NT_NY
CMDNAME c:\winnt\system32\wscript.exe
ARGS c:\programs\vbs\notify.vbs
```

Example: Run a UNIX Command

This example runs the report command under the user1 user ID on agent UNIX_NY:

```
AGENT UNIX_NY
CMDNAME /mfg/report
USER user1
```

Example: Redirect I/O and Pipe Output

This example redirects I/O and pipes output from a command. The agent runs the UNIX command ps -ef|grep cybAgent and pipes output to the /tmp/log file. The shell interpreter is invoked in the CMDNAME statement and the command is included in the ARGS statement.

```
AGENT SUN30RD
CMDNAME /usr/bin/ksh
ARGS -c "ps -ef|grep cybAgent >>/tmp/log"
USER user1
```

Example: Specify a Script Name Without the Full Path

In this example, an agent named WINAGENT runs a script named procscript.bat. The path to procscript.bat is set in the system path on the agent computer and the agent is configured to search for paths to command and script files.

```
AGENT WINAGENT
CMDNAME procscript.bat
```

COLUMNS Statement—Specify Line Width of List

The COLUMNS statement specifies the line width of the list.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
COLUMNS num
```

num

Specifies the line width of the list. The value corresponds to the SAPGUI Background Print Parameters dialog, Print settings section, Report width field.

Limits: 1-255

Note: The maximum line width of a list for viewing on the screen is 255.

Notes:

- Use the COLUMNS statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the COLUMNS statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- COLUMNS is a mandatory statement after a PRINTDEST statement.

Example: Specify the Line Width of the List

This example specifies 251 as the line width of the list for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   COLUMNS 251
```

COMMAND Statement—Specify Commands (HP Integrity NonStop Jobs)

The COMMAND statement specifies the command that you want to run on an HP Integrity NonStop system.

Supported Job Type

This statement is required for the HP Integrity NonStop job type (see page 105).

Syntax

This statement has the following formats:

```
COMMAND ~;
COMMAND ["]command-file[/run-option[,run-option...]/]["]
```

command-file

Specifies a file containing the command to run. Follows TACL file naming convention.

Limits: Case-sensitive

run-option

Specifies one of the following options for the command:

CPU *cpu-number*

Specifies the processor to run the command on.

Limits: Integer from 0 to 15

LIB *$library*

Specifies a user library file to search for external references from the command. This library is searched before a system library. Specify the system volume, subvolume, and file name.

NAME $*process-name*

Defines a process name for the command execution. If you do not define a process name, TACL generates one.

Limits: One to five characters (not including $). The first character must be alphabetic and the rest must be alphanumeric.

PRI *priority*

Defines the execution priority for the command. Commands with higher values execute first.

Limits: Integer from 1 to 199

TERM [*node-name*.]$*terminal-name*

Specifies the name of the home terminal on the HP Integrity NonStop system.

Limits: Terminal-name is one to six alphanumeric characters (not including $).

Notes:

- The first format's value '~;' is used to indicate that the job is directly invoking a command. An ARGS statement should be included in the job containing the commands to run.
- The length of the entire COMMAND statement value is limited to 256 characters.
- If you specify *run-option*, enclose the entire COMMAND statement value in double quotes, for example:

```
COMMAND "$C35.PROD.GETDEF/CPU 5, PRI 110, NAME $job1/"
```

Example: Run a Command on an HP Integrity NonStop System

In this example, the COMMAND statement runs program file $C35.PROD.GETAPD. The volume disk name is $C35, the subvolume is PROD, and the file name is GETAPD.

```
USER prod.glsys
AGENT PROAGENT
COMMAND $C35.PROD.GETAPD
ENVAR ASN1="SNAME;$VOL.SUBVOL.FILE1;PEXT=100;SEXT=500"
ENVAR PRM1="LNAME=$DATA2.LOGS.LOG1"
```

COMMAND Statement—Specify Commands to Run (Windows and UNIX Jobs)

COMMAND is an alias of CMDNAME (see page 284).

COMMANDNAME Statement—Specify Commands to Run (Windows and UNIX Jobs)

COMMANDNAME is an alias of CMDNAME (see page 284).

COMMUNITY Statement—Specify the SNMP v1 or v2 Read Community

The COMMUNITY statement specifies the SNMP v1 or v2 read community. This statement is ignored if your SNMP version is v3.

Supported Job Types

If your SNMP version is v1 or v2, this statement is required for the following job types:

- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
COMMUNITY community
```

community

Specifies the community string that is used to authenticate against the network device.

Limits: Up to 256 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify Read Community in an SNMP Value Get Job

Suppose that you want to know the value of the agentVersion variable hosted by a network device. In this example, the host name of the network device is host.example.com and its port is 161. The SNMP version is v2 and the read community is public. The name of the MIB file is RFC1213-MIB.mib, which is located on the agent computer.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE agentVersion HOST(host.example.com) PORT(161) VERSION(2)
COMMUNITY public
```

CONNECTION Statement—Specify the JNDI Name of the Connection Factory

CONNECTION is an alias of CONNECTION_FACTORY (see page 292).

CONNECTION_FACTORY Statement—Specify the JNDI Name of the Connection Factory

The CONNECTION_FACTORY statement specifies the connection factory JNDI name in a JMS job.

Supported Job Types

This statement is required for the following job types:

- JMS Publish (see page 47)
- JMS Subscribe (see page 49)

Syntax

This statement 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 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- CONNECTION is an alias of CONNECTION_FACTORY.

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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

CONNECTIONDOMAIN Statement—Specify a Domain for NTLM Connection Authentication

The CONNECTIONDOMAIN statement specifies the domain for connection authentication in an HTTP job. This statement is required for NTLM authentication.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
CONNECTIONDOMAIN origin_domain
```

origin_domain

Specifies the domain for NTLM connection authentication.

Limits: Up to 1024 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

CONNECTIONORIGIN Statement—Specify an Origin Host Name for NTLM Connection Authentication

The CONNECTIONORIGIN statement specifies the origin host name for NTLM connection authentication in an HTTP job. If you omit this statement, the origin defaults to the computer name where the agent is running.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
CONNECTIONORIGIN origin_host
```

origin_host

Specifies the origin host name for NTLM connection authentication.

Limits: Up to 1024 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

CONNECTIONUSER Statement—Specify a User Name for Connection Authentication

The CONNECTIONUSER statement specifies the user name required by a website for connection authentication and, optionally, the user's domain in an HTTP job. The scheduling manager uses the user name, domain, agent name, and the job type to select the user's password in the password repository.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
CONNECTIONUSER connect_user [DOMAIN(domain)]
```

connect_user

Specifies the user name required by a website for connection authentication.

Limits: Up to 128 characters; case-sensitive; it cannot contain delimiters (such as spaces)

DOMAIN(*domain*)

(Optional) Specifies the name of the user's domain.

Limits: Up to 64 characters; case-sensitive; it cannot contain delimiters (such as spaces)

Note: Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

Example: Specify User for Connection 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

CONTEXTENGINEID Statement—Specify a Context Engine ID

The CONTEXTENGINEID statement specifies the context engine ID, which uniquely identifies an SNMP entity within an administrative domain. An SNMP entity can access management information in many contexts. Since an item of management information can exist in more than one context, you can use the CONTEXTENGINEID statement, together with the CONTEXTNAME statement, to uniquely identify a variable within an administrative domain.

This statement is ignored if your SNMP version is v1 or v2.

Supported Job Types

This statement is optional for the following job types:

- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
CONTEXTENGINEID X'[context_engine_ID]'
```

context_engine_ID

(Optional) Specifies the context engine ID in hexadecimal format.

Limits: Up to 512 hexadecimal characters

Example: Specify Context Engine ID in an SNMP Value Set Job

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

CONTEXTNAME Statement—Specify a v3 Context Name

The CONTEXTNAME statement specifies the name of an SNMP context. An SNMP context is a collection of management information accessible by an SNMP entity. Since an item of management information can exist in more than one context, you can use the CONTEXTNAME statement, together with the CONTEXTENGINEID statement, to uniquely identify a variable within an administrative domain.

This statement is ignored if your SNMP version is v1 or v2.

Supported Job Types

This statement is optional for the following job types:

- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
CONTEXTNAME [context_name]
```

context_name

(Optional) Specifies the name of the context that the variable belongs to.

Limits: Up to 256 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify Context Name in an SNMP Value Set Job

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

COPIES Statement—Specify the Number of Copies to Print (Oracle E-Business Suite Jobs)

COPIES is an alias of PRINTCOPIES (see page 460).

COPIES Statement—Specify Number of Copies for SAP Report (SAP Jobs)

COPIES is an alias of PRINTCOPIES (see page 462).

CPU Statement—Specify Conditions to Monitor CPU Use

The CPU statement specifies the conditions that a CPU Monitoring job uses to monitor the CPU usage of the computer where the agent is installed.

Supported Job Type

This statement is optional for the CPU Monitoring job type (see page 122).

This statement does not apply when the WAITMODE statement is set to NOW.

Syntax

This statement has the following format:

```
CPU [FROM(lower)] [TO(upper)] [NOCHANGE(percent)] [CONTINUOUS(alertid)]
[WITHIN|OUTSIDE] [AVAILABLE|USED]
```

FROM(*lower*)

(Optional) Defines the lower boundary of CPU usage in percent.

Limits: 0-100

Note: If you specify the FROM operand without the TO operand, the range is between the lower boundary and 100 percent.

TO(*upper*)

(Optional) Defines the upper boundary of CPU usage in percent.

Limits: 0-100

Note: If you specify the TO operand without the FROM operand, the range is between zero percent and the upper boundary.

NOCHANGE(*percent*)

(Optional) Defines the percentage value that the CPU usage must change to trigger the alert. The trigger will not occur if the value change is within the specified percentage value.

Limits: 0-100

Notes:

- To use the NOCHANGE operand, specify an alert in the CONTINUOUS operand.
- The alert will trigger only if the following conditions are met:
  - The change value is within the range specified by the FROM and TO operands.
  - The change value is greater than the delta specified by the last scanned amount. That is, the first time CPU usage matches the job criteria, an alert is triggered. Subsequently, an alert is triggered only if the change in CPU usage is greater than the delta calculated using the last scanned amount that was registered in a trigger.

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified CPU monitoring conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified CPU monitoring conditions occur.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: When using the CONTINUOUS operand, you must also specify the FROM operand, the TO operand, or both.

WITHIN|OUTSIDE

(Optional) Specifies whether the job completes (or triggers an alert if monitoring continuously) when the value of CPU usage is within or outside the specified range.

WITHIN

Specifies that the job completes (or triggers an alert if monitoring continuously) when the value of CPU usage is within the range specified by the FROM and TO operands. This is the default.

OUTSIDE

Specifies that the job completes (or triggers an alert if monitoring continuously) when the value of CPU usage is outside the range specified by the FROM and TO operands.

AVAILABLE|USED

(Optional) Specifies whether the job monitors the available or used CPU capacity.

AVAILABLE

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.

- **Note:** 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: Continuously Monitor and Report CPU Usage Outside a Range

This example continuously monitors the CPU available on the SYSAGENT computer. An alert is triggered each time the available CPU is less than 20 percent or greater than 50 percent, and the CPU usage changes by more than 10 percent. If the percent change is less than or equal to 10 percent, the job does not register a change.

```
AGENT SYSAGENT
CPU FROM(20) TO(50) NOCHANGE(10) CONTINUOUS(al3) +
  OUTSIDE AVAILABLE
```

CREATEMETHOD Statement—Specify a Create Method

The CREATEMETHOD statement specifies the name of the create method in an Entity Bean and Session Bean job.

Supported Job Types

This statement is optional for the following job types:

- Entity Bean (see page 39)
- Session Bean (see page 65)

Note: The create method is required to access a stateful session bean.

Syntax

This statement has the following format:

```
CREATEMETHOD create_method
```

create_method

Specifies the name of the create method.

Default: create

Limits: Up to 1024 characters; case-sensitive; it must always begin with create

Example: createaccount

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
CREATEMETHOD createcustomer
OPERATIONTYPE CREATE
CREATEPARAMETER TYPE(String) VALUE(customerid)
CREATEPARAMETER TYPE(String) VALUE(800-555-0100)
```

CREATEPARAMETER Statement—Specify Create Parameters

The CREATEPARAMETER statement 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.

Supported Job Types

This statement is optional for the following workload objects:

- Entity Bean (see page 39)
- Session Bean (see page 65)

Note: To access stateful session beans, you must specify create parameters.

Syntax

This statement has the following format:

```
CREATEPARAMETER {TYPE(type) VALUE(value)}
                {TYPE(type) ARRAY(value,value,...)}
                {URI(uri)}
```

TYPE(*type*)

Specifies the Java class of the parameter.

Limits: Up to 1024 characters; case-sensitive

Examples: String, java.lang.String, Integer

Note: If the package is not specified, java.lang is assumed.

Value(*value*)

Specifies the String value for the method parameter.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of string values for the method parameter.

Limits: Up to 1024 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple parameters, use multiple CREATEPARAMETER statements. Order is important. For example, if the method contains three parameters, specify the first parameter in the first statement, the second parameter in the second statement, and the third parameter in the third statement. 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN Shoppingcart
CREATEMETHOD createaddbook
METHOD addbook
CREATEPARAMETER TYPE(String) VALUE(ISBN)
PARAMETER TYPE(Integer) ARRAY(1551929120,1582701709)
```

CURLIB Statement—Specify the Name of the Current Library

The CURLIB statement 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 statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
CURLIB current_libraryc|
       /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.

```
AGENT I5AGENT
CLPNAME MFGDATA
CURLIB PAY1
JOBD CYB1/DFTJOBD
```

CUSTOMPROP Statement—Specify Value Set Expressions for Default Values

The CUSTOMPROP statement specifies value set expressions for default values. For example, if the default value of an argument is an SQL statement that contains profile (:$PROFILES$) or flexfield (:$FLEX$) expressions, you can specify how the agent resolves the expressions using this statement.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
CUSTOMPROP KEY(expression) VALUE(value)
```

expression

Specifies the expression to resolve.

Limits: Up to 256 characters; case-sensitive

Examples:

- :$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT
- :$PROFILES$.GL_ACCESS_SET_ID

value

Specifies the default value for the expression. The value can contain a constant or an SQL SELECT query to run.

Limits: Up to 256 characters; case-sensitive

Examples:

- SELECT EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT from SOME_TABLE where SOME_KEY = ‘Default’
- SELECT '1' FROM DUAL

Notes:

- If you want the agent to quote the resolved value, put the default value in single quotes, for example, 'SELECT '1' FROM DUAL'.
- The agent does not resolve Field or Segment validation value sets.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple default value set expressions, use multiple CUSTOMPROP statements.

Example: Specify a Default Value Set Expression

Suppose that the default value of an argument in a single request program is the following SQL statement:

```
SELECT inventory_item_id from mtl_system_items where SEGMENT1 =
:$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT
```

To resolve the flexfield expression, :$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT, you define the following default value set expression in the job:

```
:$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT='SELECT
EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT from SOME_TABLE where SOME_KEY = 'Default''
```

The default value for the expression is enclosed in single quotes so that the agent quotes the resolved value.

Assuming that the SQL SELECT statement returns Acct123, the following SQL query determines the default argument value:

```
SELECT inventory_item_id from mtl_system_items where SEGMENT1 = 'Acct123'
```

The following job definition specifies the default value set expression:

```
AGENT CYBOA
PROGRAM GMFDSUR
APPLDISPLNAME 'Process Manufacturing Financials'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
CUSTOMPROP KEY(':$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT') +
  VALUE('SELECT EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT +
         from SOME_TABLE where SOME_KEY = 'Default'')
```

DB_URL Statement—Specify Database Resource Location

The DB_URL statement specifies the database resource location for a database job.

Supported Job Types

This statement is optional for the following job types:

- Database Monitor (see page 69)
- Database Stored Procedure (see page 71)
- Database Trigger (see page 76)
- SQL (see page 81)

Syntax

This statement has the following format:

```
DB_URL url
```

url

Specifies a JDBC database resource location for a job. Use the appropriate format for your database type:

- For an Oracle database:

  ```
  DB_URL jdbc:oracle:thin:@host:port:dbname
  ```

- For a Microsoft SQL Server database:

  ```
  DB_URL jbdc:sqlserver://host:port;DatabaseName=dbname
  ```

- For an IBM DB2 database:

  ```
  DB_URL jdbc:db2://host:port/dbname
  ```

Limits: Up to 256 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you do not specify this statement, a default database resource location must be defined in the db.default.url parameter in the agent's agentparm.txt file.
- The DB_URL statement overrides the default resource location specified in the agent's agentparm.txt.
- HOST is an alias of DB_URL.

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.

```
AGENT CYBDB1
SQL 'SELECT * from NEWORDS'
DB_URL jdbc:oracle:thin:@172.31.255.255:1433:ORDERS
```

Example: Specify a Microsoft SQL Server Database

This example uses the MS SQL Server database named ORDERS that is reachable on port 1433 on a host named myhost.

```
AGENT CYBDB1
SQL 'SELECT * from NEWORDS'
DB_URL '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.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://172.31.255.255:50000/SAMPLE
SQL 'UPDATE ENTADM.STAFF SET YEARS=3 where NAME=''Jonson'''
```

DEFINE Statement—Define a Logical Name for a File-System Element

The DEFINE statement enables you to define a logical name for a file-system element such as a tape drive, spooler subsystem, and a volume.subvolume location.

Supported Job Types

This statement is optional for the HP Integrity NonStop job type (see page 105).

Syntax

This statement has the following format:

```
DEFINE "element-name;CLASS=class;class-attribute;class-attribute..."
```

The CLASS=*class* and *class-attribute* operands have the following format:

```
{CLASS=MAP;FILE=file-name}
{CLASS=SEARCH
   [;RELSUBVOLn=subvolume-name]
   [;SUBVOLn=subvolume-name]}
{CLASS=SORT
   [;BLOCK=size]
   [;CPU=cpu-number]|[;CPUS=ALL|cpu-number,cpu-number...]
   [;MODE=AUTOMATIC|MINSPACE|MINTIME]
   [;NOTCPUS=cpu-number,cpu-number...]
   [;PRI=priority]
   [;PROGRAM=file-name]
   [;SCRATCH=file-name]
   [;SEGMENT=size]
   [;SUBSORTS=define-name,define-name...]
   [;SWAP=file-name]}
{CLASS=TAPE
   [;BLOCKLEN=length]
   [;DENSITY=800|1600|6250]
   [;RETENTION=days]
   [;SYSTEM=\node-name]
   [;TAPEMODE=STARTSTOP|STREAM]
   [;USE=IN|OUT|EXTEND|OPENFLAG]
   [;VERSION=version]
   [;VOLUME=vol-id|SCRATCH]
   [;DEVICE=device-name,device-name...]
   [;REELS=volume]
   [;EBCDIC=IN|OUT|ON|OFF]
   [;EXPIRATION=date]
   [;FILEID=file-name]
   [;FILESECT=vol1,vol2...]
   [;FILESEQ=file1,file2...]
   [;GEN=gen-num]
   [;LABELS=ANSI|IBM|OMITTED|BYPASS|BACKUP|IBMBACKUP]
   [;MOUNTMSG="message-text"]
   [;OWNER=owner-id]
   [;RECFORM=F|U]
   [;RECLEN=record-length]}
```

element-name

Specifies the name of the file-system element.

Limits: No less than two characters and no more than 24 characters. Case-sensitive. Must start with a letter or equal sign. After which, you can use the following characters: alphanumeric, hyphen, underscores, semicolons, and circumflex (up caret).

class

Specifies the subtype of definition being created. The classes are as follows:

MAP

Specifies a correlation between a logical device and an actual file.

SEARCH

Specifies a search list of subvolumes for a program. The SEARCH class functions similarly to the TACL #PMSEARCHLIST built-in variable.

SORT

Specifies parameters for FastSort processes and parallel SORTPROG processes.

TAPE

Used for accessing labeled tapes.

class-attribute

Specifies an attribute associated with a particular value of the *class* operand.

Note: The attributes are described in the SET DEFINE command, which is included in the *HP NonStop TACL Reference Manual*, available from HP.

Limits: You can use the following characters in the value of a class attribute: alphanumeric, hyphen, underscores, and circumflex (up caret).

Notes:

- The text string in double quotes in the DEFINE statement cannot be longer than 1024 characters.
- The DEFINE statement is case-sensitive and it can contain the delimiters comma, semicolon, and space.
- The combined length of all DEFINE statement text strings in a job cannot exceed 4096 characters minus the number of entities coded. For example, if 64 DEFINE statements are coded, the maximum length for all DEFINE data is 4096 – 64.

Example: Use the DEFINE Statement

In this example, the DEFINE statement defines logical name MYFIL1, which is used in place of the file name $DATA2.TCSDEMO.FINFOJOB.

```
AGENT PROAGENT
USER prod.glsys
COMMAND $C35.TCSOBJ.OABC
ENVAR STDOUT=$C35.TCSOBJ.cafout1
ENVAR STDERR=$c35.srkobj.caferr1
ASSIGN "$VOL.SUBVOL.FILE1;$VOL1.SUBVOL1.FILE2"
PARAMETER ABC(100)
DEFINE "MYFIL1;CLASS=MAP;FILE=$DATA2.TCSDEMO.FINFOJOB"
```

DESC Statement—Describe an Oracle Applications Single Request

DESC is an alias of DESCRIPTION (see page 312).

DESCRIPTION Statement—Describe an Oracle Applications Single Request

The DESCRIPTION statement provides an additional description for an Oracle Applications single request. This description is concatenated with the program name to create the request name.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Single Request job type (see page 142).

Syntax

This statement has the following format:

```
DESCRIPTION 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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can define 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 DESCRIPTION statement overrides the default description specified in the oa.default.desc parameter in the agent's agentparm.txt.
- DESC is an alias of DESCRIPTION.

Example: Add a Description to a Single Request

In the following example, information on the history of a program is included in the description:

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
DESCRIPTION 'CONFIRMED REQ 20040808 by TYB212'
```

DEST Statement—Specify an Output Destination File

DEST is an alias of DESTINATION (see page 314).

DESTINATION Statement—Specify an Output Destination File

The DESTINATION statement specifies the output destination file in a job. In a payload producing job, the file stores the Java serialized object produced by the job.

Note: If you omit this statement, 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.

Supported Job Types

This statement is optional for the following job types:

- JMS Subscribe (see page 49)
- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Operation (see page 56)
- POJO (see page 60)
- RMI (see page 62)
- Session Bean (see page 65)
- SNMP Subscribe (see page 178)
- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)
- Web Service (see page 211)

Syntax

This statement has the following format:

`DESTINATION` *`destination`*

destination

Specifies the output destination file.

Limits: Up to 256 characters; case-sensitive

Examples: /tmp/myoutput, 'D:\Stock output\STOCK3'

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- DEST is an alias of DESTINATION.

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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(a23)
```

DESTINATIONNAME Statement—Specify the JNDI Name of the Topic or Queue

DESTINATIONNAME is an alias of DESTNAME (see page 316).

DESTNAME Statement—Specify the JNDI Name of the Topic or Queue

The DESTNAME statement 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 statement is required for the following workload objects:

- JMS Publish (see page 47)
- JMS Subscribe (see page 49)

Syntax

This statement has the following format:

- For the JMS Publish job type:

  `DESTNAME` *`destination`*

- For the JMS Subscribe job type:

  `DESTNAME` *`destination`* `[CONTINUOUS(`*`alertid`*`)]`

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

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when a message matches the filter condition in a JMS Subscribe job. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the agent detects the filter value. The agent ignores all messages that do not match the filter.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- DESTINATIONNAME is an alias of DESTNAME.

Example: Publish a Message to a Queue on 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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

Example: Monitor a Queue on a WebLogic Application Server

This example continuously monitors the queue named Queue (residing on WebLogic) for messages 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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://172.24.0.0:7001
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue CONTINUOUS(a13)
FILTER abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+
TOPIC N
DESTINATION /export/home/user1/outputfile1
JNDIUSER 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

DIRECTION Statement—Specify the Transfer Direction

DIRECTION is an alias of TRANSFERDIRECTION (see page 625).

DISABLE_RESTART Statement—Specify Whether to Disable the Restart Feature for Failed PeopleSoft Jobs

The DISABLE_RESTART statement specifies whether to disable a restart feature for previously failed PeopleSoft jobs. By default, the restart feature is enabled. When a previously failed job is resubmitted, it restarts from where it stopped.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
DISABLE_RESTART YES|NO
```

YES

Disables the restart feature. Previously-failed jobs restart from the beginning.

NO

Does not disable the restart feature. When a failed job is resubmitted, it restarts from where it stopped. This is the default.

Notes:

- The restart feature only applies to certain PeopleSoft Process Types such as Application Engine.
- DR is an alias of DISABLE_RESTART.

Example: Disable the Restart Feature for Failed PeopleSoft Jobs

This example disables the restart feature. Previously failed jobs restart from the beginning.

```
AGENT PSAGT
PROCESSNAME XRFWIN
PROCESSTYPE 'Application Engine'
DISABLE_RESTART YES
```

DISK Statement—Specify Conditions to Monitor Disk Space

The DISK statement specifies the conditions that a Disk Monitoring job uses to monitor the disk space of the computer where the agent is installed.

Supported Job Type

This statement is required for the Disk Monitoring job type (see page 123).

Syntax

This statement has the following format:

```
DISK drive [FORMAT(PERCENT|GB|MB|KB|B)]
   [FROM(lower)] [TO(upper)] [NOCHANGE(value)]
   [CONTINUOUS(alertid)][WITHIN|OUTSIDE]
   [AVAILABLE|USED]
```

drive

Specifies the path to the disk, logical partition, or auxiliary storage pool to be monitored.

Limits: Up to 256 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`*`digit1digit2`*

FORMAT(PERCENT|GB|MB|KB|B)

(Optional) Specifies the unit of measurement used to monitor available or used disk space. Options are the following:

PERCENT

Monitors disk usage by percentage.

Note: If you specify PERCENT, the values specified by the FROM and TO operands cannot 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. This is the default.

FROM(*lower*)

(Optional) Defines the lower boundary of disk usage. The unit for this value is specified in the FORMAT operand.

Limits:

- 0-100 (This limit applies when the disk format is PERCENT.)
- Up to 10 digits (This limit applies for all other disk formats.)

Example: 35000000

Note: If you specify the FROM operand without the TO operand, the range is between the lower limit and the maximum available disk space.

TO(*upper*)

(Optional) Defines the upper boundary of disk usage. The unit for this value is specified in the FORMAT operand.

Limits:

- 0-100 (This limit applies when the disk format is PERCENT.)
- Up to 10 digits (This limit applies for all other disk formats.)

Example: 36000000

Note: If you specify the TO operand without the FROM operand, the range is between zero and the upper boundary.

NOCHANGE(*value*)

(Optional) Defines the amount of change that the disk usage must change to trigger the alert. The unit for this value is specified in the FORMAT operand.

Limits: Up to 10 digits

Example: 100

Notes:

- To use the NOCHANGE operand, specify an alert in the CONTINUOUS operand.
- The alert will trigger only if the following conditions are met:
  - The change value is within the range specified by the FROM and TO operands.
  - The change value is greater than the delta specified by the last scanned amount. That is, the first time disk usage matches the job criteria, an alert is triggered. Subsequently, an alert is triggered only if the change in disk usage is greater than the delta calculated using the last scanned amount that was registered in a trigger.

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified disk monitoring conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified disk monitoring conditions occur.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: When using the CONTINUOUS operand, you must also specify the FROM operand, the TO operand, or both.

WITHIN|OUTSIDE

(Optional) Specifies whether the job completes (or triggers an alert if monitoring continuously) when the value of disk usage is within or outside the specified range.

WITHIN

Specifies that the job completes (or triggers an alert if monitoring continuously) when the value of disk usage is within the range specified by the FROM and TO operands. This is the default.

OUTSIDE

Specifies that the job completes (or triggers an alert if monitoring continuously) when the value of disk usage is outside the range specified by the FROM and TO operands.

AVAILABLE|USED

(Optional) Specifies whether the job monitors available or used disk space.

AVAILABLE

Specifies that the job reads the available disk space for monitoring. This is the default.

USED

Specifies that the job reads the used disk space for monitoring.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- The FROM, TO, NOCHANGE, CONTINUOUS, WITHIN, OUTSIDE, AVAILABLE, and USED operands do not apply when the WAITMODE statement is set to NOW.

Example: Monitor Available Space on a UNIX Partition

This example monitors a local UNIX partition for available space. The job completes immediately and reports the available space in megabytes.

```
AGENT UNIXAGENT
DISK /export/home FORMAT(MB)
```

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 named DISK is issued. The job continues monitoring the disk space until it is ended manually.

```
AGENT UNIXAGENT
DISK /export/home FORMAT(PERCENT) FROM(90) USED +
    CONTINUOUS(DISK)
```

Example: Continuously Monitor Used Space on a Windows Drive

This example continuously monitors the C drive on a Windows computer for used space. Each time the used disk space falls below 16 percent or exceeds 95 percent, an alert named DISK is triggered. The job continues monitoring the disk space until it is ended manually.

```
AGENT WINAGENT
DISK C FORMAT(PERCENT) FROM(16) TO(95) USED OUTSIDE +
    CONTINUOUS(DISK)
```

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 triggered.

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.

```
AGENT WINAGENT
DISK C FORMAT(KB) FROM(35000000) TO(36000000) NOCHANGE(100)+
    CONTINUOUS(disk)
```

The following table shows four sequential scans:

Scan	Scanned Amount (Kilobytes)	Does the trigger occur?
1	35018896	Yes.

Scan	Scanned Amount (Kilobytes)	Does the trigger occur?
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 range specified by the FROM and TO operands.

Example: Monitor the System Auxiliary Storage Pool on an i5/OS System

This example continuously monitors the system auxiliary storage pool for used disk space. Each time the used disk space falls between 90 and 100 percent, an alert named DISK is triggered. The job continues monitoring the disk space until it is ended manually.

```
AGENT I5AGENT
DISK / FORMAT(PERCENT) FROM(90) TO(100) USED CONTINUOUS(DISK)
```

DISPLNAME Statement—Specify the Display Name of an Oracle Applications Application

DISPLNAME is an alias of APPLDISPLNAME (see page 227).

DISTRFOLDER Statement—Specify the Name of a PeopleSoft Distribution Detail Folder

The DISTRFOLDER statement specifies the name of a report folder for the contents of a distribution list.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
DISTRFOLDER 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.

```
AGENT PSAGENT
PROCESSTYPE 'Application Engine'
OUTDESTTYPE WEB
PROCESSNAME AEMINITEST
OUTDESTFORMAT PTF
PSOPRID VP1
RUNCONTROLID test
SERVERNAME PSNT
DISTRFOLDER GENERAL
```

DISTRLISTROLES Statement—Specify a Distribution List of Roles

The DISTRLISTROLES statement specifies a distribution list of the roles that represent the individuals who are receiving the PeopleSoft report.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
DISTRLISTROLES role|'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:

- If you specify a single role containing spaces, enclose it in single quotation marks.
- You can specify multiple roles. Separate each role with a comma and enclose the entire list of roles in single quotes.
- The entire value can be up to 4078 characters.
- An alternative to using this statement is to use operator IDs as specified by the DISTRLISTUSERS statement.
- DISTRROLES is an alias of DISTRLISTROLES.

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.

```
AGENT PSAGENT
PROCESSTYPE Crystal
PROCESSNAME XRFWIN
OUTDESTTYPE WEB
OUTDESTFORMAT PDF
DISTRLISTROLES 'CLERK,BANK MANAGER,Employee'
PSOPRID VP1
```

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.

```
AGENT PSAGENT
PROCESSTYPE Crystal
PROCESSNAME XRFWIN
OUTDESTTYPE WEB
OUTDESTFORMAT PDF
DISTRLISTROLES 'BANK MANAGER'
PSOPRID VP1
```

DISTRLISTUSERS Statement—Specify a Distribution List of Operator IDs

The DISTRLISTUSERS statement specifies a distribution list of operator IDs to send a PeopleSoft report to.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
DISTRLISTUSERS operator_id|'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

Notes:

- If you specify a single operator ID containing spaces, enclose it in single quotation marks.
- You can specify multiple operator IDs. Separate each operator ID with a comma and enclose the entire list of operator IDs in single quotes.
- The entire value can be up to 4078 characters.
- By default, the report output is sent to the operator ID PSOPRID. If you do not want to distribute the output to any operator IDs, you must include a DISTRLISTUSERS statement 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 DISTRLISTROLES statement.
- If the output destination type is EMAIL or WEB, you can also distribute the report output to a list of email addresses using the EMAILADDR statement.
- DISTRUSERS is an alias of DISTRLISTUSERS.

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.

```
AGENT PSAGENT
PROCESSTYPE Crystal
PROCESSNAME XRFWIN
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP3
RUNCONTROLID test
DISTRLISTUSERS 'VP1,VP2,VP3'
```

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 DISTRLISTUSERS statement has no value, so the output is sent to the email addresses specified in the EMAILADDR statement. The email includes a subject title and body text. The web report is included with the email.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
DISTRLISTUSERS
EMAILWEBREPORT Y
OUTDESTTYPE WEB
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

DISTRROLES Statement—Specify a Distribution List of Roles

DISTRROLES is an alias of DISTRLISTROLES (see page 325).

DISTRUSERS Statement—Specify a Distribution List of Operator IDs

DISTRUSERS is an alias of DISTRLISTUSERS (see page 327).

DR Statement—Specify Whether to Disable the Restart Feature for Failed PeopleSoft Jobs

DR is an alias of DISABLE_RESTART (see page 318).

EMAILADDR Statement—Specify the Email Addresses on a Distribution List (PeopleSoft Jobs)

The EMAILADDR statement specifies the email addresses of the recipients on a distribution list. The PeopleSoft report is emailed to the recipients.

Note: To use this statement, you must specify EMAIL or WEB in the OUTDESTTYPE statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
EMAILADDR 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 and enclose the entire list of addresses in single quotes.
- 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 EMAILADDR statement. The email includes a subject title and body text.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

EMAILADDR Statement—Email Spool File on ABAP Completion or Failure (SAP Jobs)

The EMAILADDR statement emails a copy of the SAP spool file to one or more email addresses upon ABAP completion or failure.

Supported Job Type

This statement is optional for the SAP R/3 (see page 166) job type.

Syntax

This statement has the following format:

```
EMAILADDR address |'address,address,...'
```

address

Specifies the email address of a recipient on a distribution list.

Limits: Up to 256 valid SAP characters; case-sensitive

Note: To specify multiple email addresses, separate each address with a comma and enclose the entire list of addresses in single quotation marks.

Notes:

- Use the EMAILADDR statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the EMAILADDR statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- You can also use the SAPEMAILADDR statement to email the spool file upon job completion or failure.

Example: Email the Spool File to Two Addresses on Step Completion or Failure

This example emails the spool output to the user1@example.com and user2@example.com addresses when the BTCTEST ABAP completes or fails:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   EMAILADDR 'user1@example.com,user2@example.com'
```

EMAILLOG Statement—Specify Whether to Email PeopleSoft Job Logs

The EMAILLOG statement specifies whether to email job logs along with the PeopleSoft report to recipients on a distribution list.

Note: To use this statement, you must specify EMAIL or WEB in the OUTDESTTYPE statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
EMAILLOG 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 statement corresponds to the Email With Log field in PeopleSoft.

Example: Email PeopleSoft Job Logs

This example runs a Crystal report under the VP2 operator ID. The output is stored as a PDF web report. The DISTRLISTUSERS statement has no value, so the output is sent to the email addresses specified in the EMAILADDR statement. The email includes a subject title and body text. The web report and the job logs are included with the email.

```
AGENT PSAGT
PROCESSNAME XRFWIN
PROCESSTYPE 'Crystal reports'
OUTDESTTYPE WEB
OUTDESTFORMAT PDF
DISTRLISTUSERS
PSOPRID VP2
EMAILWEBREPORT YES
EMAILADDR 'user1@example.com;user2@example.com'
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'This report is available for distribution.'
EMAILLOG YES
```

EMAILSUBJECT Statement—Define an Email Subject

The EMAILSUBJECT statement defines an email subject to include in the email to distribute to recipients of a PeopleSoft report.

Note: To use this statement, you must specify EMAIL or WEB in the OUTDESTTYPE statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
EMAILSUBJECT 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

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 EMAILADDR statement. The email includes a subject title and body text.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

EMAILTEXT Statement—Define the Body Text of an Email

The EMAILTEXT statement defines the body text of the email to distribute to recipients of a PeopleSoft report.

Note: To use this statement, you must specify EMAIL or WEB in the OUTDESTTYPE statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
EMAILTEXT email_text
```

email_text

Defines the body text of the email. This value corresponds to the Message Text field in PeopleSoft.

Limits: Up to 1024 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 EMAILADDR statement. The email includes a subject title and body text.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

EMAILWEBREPORT Statement—Specify Whether to Email a PeopleSoft Web Report

The EMAILWEBREPORT statement specifies whether to email a web report to recipients on a distribution list.

Note: To use this statement, you must specify WEB in the OUTDESTYPE statement. You must also specify one or more addresses using the EMAILADDR statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
EMAILWEBREPORT 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 statement 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 EMAILADDR statement. The email includes a subject title and body text.

```
AGENT PSAGENT
EMAILADDR 'user1@example.com;user2@example.com'
EMAILWEBREPORT YES
OUTDESTTYPE WEB
OUTDESTFORMAT PDF
PSOPRID VP2
EMAILSUBJECT 'PeopleSoft Report Status'
EMAILTEXT 'The report is available for distribution'
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
```

ENCODING Statement—Specify the Character Encoding of a File

The ENCODING statement specifies the character encoding of the text file to be monitored. You can omit this statement if the text file is ASCII.

Supported Job Type

This statement is optional for the Text File Reading and Monitoring job type (see page 127).

Syntax

This statement 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.

Limits: Up to 256 characters; case-sensitive

Examples: US-ASCII, ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16

Note: The supported character sets depends on your operating system and Java Virtual Machine (JVM).

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Monitor a Text File Encoded in ISO-8859-1

The following job monitors a text file that contains data encoded in ISO-8859-1 (ISO Latin Alphabet No. 1 or ISO-LATIN-1). The job completes successfully if the specified string is found.

```
AGENT MONAGT
TEXTFILE /export/home/logs/transactions.log
TEXTSTRING 'ERROR MESSAGE'
SEARCHRANGE LINE FROM(1) TO(50)
ENCODING ISO-8859-1
```

ENDPOINT_URL Statement—Specify a Target Endpoint URL

The ENDPOINT_URL statement specifies the target endpoint address URL in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement 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 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- In a Web Service job, if you specify the WSDL_URL statement but not the ENDPOINT_URL statement, you must specify both the SERVICENAME and PORTNAME statements.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

ENGINEID Statement—Specify an Engine ID

The ENGINEID statement specifies the SNMP engine ID, which uniquely identifies an SNMP engine and its corresponding SNMP entity within an administrative domain. By default, SNMP trap information is sent from the default agent engine ID.

This statement is ignored if your SNMP version is v1 or v2.

Supported Job Type

This statement is optional for the SNMP Trap Send job type (see page 179).

Syntax

This statement has the following format:

```
ENGINEID engine_ID
```

engine_ID

(Optional) Specifies the SNMP engine ID.

Limits: Up to 256 characters; case-sensitive

Default: AGENT_ENGINE

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the Default Engine ID

Suppose that you want to send the cybtrapstart trap to a network device using SNMP v3. In this example, five string parameters are passed to the trap. The host name of the network device is localhost and its port is 162. The job uses the AES privacy protocol and the SHA authentication protocol. The name of the MIB file is RFC1213-MIB.mib and the default engine ID is used. The credentials of user1 are used for authorization.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE cybtrapstart HOST(localhost) PORT(162) VERSION(3)
SNMPUSER user1
AUTHPROTOCOL SHA
PRIVPROTOCOL AES
ENGINEID
PARAMETER TYPE(snmp:string) VALUE(p1)
PARAMETER TYPE(snmp:string) VALUE(p2)
PARAMETER TYPE(snmp:string) VALUE(p3)
PARAMETER TYPE(snmp:string) VALUE(p4)
PARAMETER TYPE(snmp:string) VALUE(p5)
```

ENVAR Statement—Pass Environment Variables

The ENVAR statement passes environment variables to a batch file, script, command, or program.

Supported Job Types

This statement is optional for the following job types:

- HP Integrity NonStop (see page 105)
- PeopleSoft (see page 146)
- UNIX (see page 190)
- Windows (see page 214)

Syntax

This statement has the following format:

```
ENVAR name=value |
      EWA_ENV_SRC_N=path |
      EWAGLOBALPROFILE=path |
      HOME=path | PWD=path |
      STDIN=keyboard|path | STDOUT=spoolfile|path | STDERR=spoolfile|path
```

name=value

Specifies the name of the environment variable and the value assigned to the variable.

Note: If the resulting value requires spaces, such as a date, enclose the value in double quotes in the ENVAR statement.

EWA_ENV_SRC_*N*=*path*

Specifies the full path of the file or script to be sourced for environment variables, where *N* is a positive integer. You can use this variable in UNIX and Windows jobs to pass environment variables set in a file or script prior to job execution.

UNIX:

- To use this variable, the agent must run as root and the oscomponent.loginshell agent parameter must be set to true. For more information about the oscomponent.loginshell agent parameter, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- The file to be sourced must be a script that sets environment variables.

Windows: The file to be sourced must contain *name*=*value* pairs.

EWAGLOBALPROFILE=*path*

Specifies the full path of the script to be sourced for environment variables. You can use this variable in UNIX jobs to pass environment variables set in a script prior to job execution.

Notes:

- The EWAGLOBALPROFILE variable applies to UNIX jobs only. To use this variable, the agent must run as root and the oscomponent.loginshell agent parameter must be set to true.
- The file to be sourced must be a script that sets environment variables.
- The EWAGLOBALPROFILE variable can also be specified in a variables file set in the agentparm.txt file. Specifying EWAGLOBALPROFILE in a variables file is useful if you want the script to be sourced for *all* jobs run on the agent, on a specific scheduling manager, or under a specific user. In addition to the preceding requirements, the oscomponent.lookupcommand agent parameter must also be set to true. For more information on using EWAGLOBALPROFILE in a variables file, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

HOME=*path*

Specifies the fully-qualified name of the home directory to switch to before running the batch file, script, command, or program. The HOME operand applies to all job definitions except for UNIX-type jobs and Micro Focus jobs. Alternatively, you can use the SCRIPT option in the oscomponent.initialworkingdirectory parameter in the agent's agentparm.txt file.

Note: If HOME is not specified, the command or script runs in the initial working directory as specified by the oscomponent.initialworkingdirectory parameter in the agentparm.txt file. For more information about oscomponent.initialworkingdirectory, see the *CA Workload Automation Agent for UNIX, Linux, or Windows User Guide*.

PWD=*path*

Specifies the name of the Present Working Directory (PWD) to switch to before running a UNIX command or script. The PWD operand applies to UNIX job definitions only.

Note: If the PWD is not specified, the command or script runs in the initial working directory as specified by the oscomponent.initialworkingdirectory parameter in the agentparm.txt file. For more information about oscomponent.initialworkingdirectory, see the *CA Workload Automation Agent for UNIX, Linux, or Windows User Guide*.

STDIN=keyboard|*path*

Specifies the full path of an alternative standard input stream. If you do not specify the STDIN operand, by default the standard input stream is the keyboard.

STDOUT=spoolfile|*path*

Specifies the full path of an alternative standard output stream. If you do not specify the STDOUT operand, by default the standard output stream goes to the spool file.

The STDOUT file name displays when you list the workload object with the LISTAPPL or APPLJOB command. The STDOUT file name can be different from what was specified in STDOUT=. In the following example, if the environment variable AGENTPATH equaled '/root/home/2, the file name would resolve to StdOut=/root/home/2/stdout.txt in the output of the workload object.

```
ENVAR STDOUT=$AGENTPATH/stdout.txt
```

For HP Integrity NonStop, STDOUT specifies the job output file location. New output overwrites existing output unless you add the >> prefix before the file name. The >> prefix causes the adding of new output to the current output file. In the following example, any output that the job generates is added to the end of the $TEST.Test1.Output1 file.

```
ENVAR STDOUT=>>$TEST.Test1.Output1
```

STDOUT is applicable only for Windows NT, all UNIX agents, and HP Integrity NonStop.

STDERR=spoolfile|*path*

Specifies the full path of an alternative standard error output stream. If you do not specify the STDERR operand, by default the standard error output stream goes to the spool file.

The STDERR file name displays when you list the workload object with the LISTAPPL or APPLJOB command.

The STDERR file name can be different from what was specified in STDERR=. In the following example, if the environment variable AGENTPATH equaled '/root/home/2', the file name would resolve to StdErr=/root/home/2/stderr.txt in the output of the workload object.

```
ENVAR STDERR=$AGENTPATH/stderr.txt
```

For HP Integrity NonStop, STDERR specifies the job error file location. New errors overwrite existing errors unless you add the >>prefix before the file name. The >> prefix causes the adding of new errors to the current error file. In the following example, any errors that the job generates are added to the end of the $TEST.Test1.Error1 file.

```
ENVAR STDERR=>>$TEST.Test1.Error1
```

STDERR is applicable only for Windows NT, all UNIX agents, and HP Integrity NonStop. STDERR is only available for compiled binary commands. TACL scripts do not support STDERR streams.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
  - For OpenVMS, to preserve spaces, tabs, and lowercase characters in the symbol value, enclose the value in three sets of quotation marks. You must also use three sets of quotation marks to include a quotation mark as part of a string. In order to input three sets of quotation marks in the ENVAR statement, the second and third double quotation marks have to be escaped by a backslash. For example, to define the symbol TEST with value “TEST space” and to preserve the spaces and lower case characters, the ENVAR statement would be TEST=”\”\”TEST space \”\””.
- The entire value can be up to 4078 characters.
- You can specify multiple environment variables using multiple ENVAR statements. There is no limit to the number of ENVAR statements that you can use.
- Environment variables can also be specified in a variables file set in the agentparm.txt file. Environment variables specified in a variables file can apply to all jobs run on the agent, all jobs run on a specific scheduling manager, or all jobs run under a specific user. For more information about specifying environment variables in a variables file, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.*
- The environment variables specified through the ENVAR statement override the environment variables specified through the agentparm.txt file.

- 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, specify the variable name with a leading and trailing dollar sign ($).
  - For Windows batch files or programs to access environment variables, specify the variable name with a leading and trailing percent sign (%).
- The ENVAR statement 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
  ```
- 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: Pass Windows Environment Variables to a Batch File

This example includes three ENVAR statements that pass environment variables to a Windows batch file and a fourth ENVAR statement that defines the home directory:

```
CMDNAME c:\payroll\command1
AGENT NT_NY
ENVAR A=B
ENVAR C='X Y'
ENVAR E=pay
ENVAR HOME=c:\export\u1
```

In this example, the command command1 can reference these variables:

Environment Variable	Value Passed
A	B
C	'X Y'
E	pay
HOME	c:\export\u1

Example: Pass UNIX Environment Variables to a Script

This example includes two ENVAR statements that pass environment variables to a script and a third ENVAR statement that defines the Present Working Directory (PWD). The parameter "user 1" is enclosed with double quotation marks because it contains a space.

```
SCRIPTNAME /home/scripts/pay
AGENT UNIX_NY
ENVAR NAME="user 1"
ENVAR JOB=PAYROLL
ENVAR 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

Example: Source Windows Environment Variables from a File

This example uses the EWA_ENV_SRC_1 environment variable to source environment variables from a file. Before the test_var3 command file runs, the test_var4.txt file is sourced and the variables in the file are resolved.

```
AGENT WINAGT
CMDNAME %scripts\test_var3
ENVAR EWA_ENV_SRC_1=c:\qatest\env\test_var4.txt
```

The test_var4.txt file contains the following:

```
AD_TOP=C:\oracle\OA12\apps\apps_st\appl\ad\12.0.0
ADJREOPTS=-ms128m -mx256m -Xrs

NLS_LANG=American_America.WE8MSWIN1252

ZX_TOP=C:\oracle\OA12\apps\apps_st\appl\zx\12.0.0

TEST1=test1
TEST2=test two
TEST3= test 3
```

Example: Source UNIX Environment Variables from a Script Using EWA_ENV_SRC_*N*

This example uses the EWA_ENV_SRC_1 environment variable to source environment variables from a script. The agent runs as root and the oscomponent.loginshell agent parameter is set to true. Before the test_var3 script runs, the test_var_ewal.txt script is sourced and the variables in the file are resolved.

```
AGENT UNIXAGT
SCRIPTNAME %scripts/test_var3
ENVAR EWA_ENV_SRC_1=%var_dir/test_var_ewa1.txt
```

The test_var_ewal.txt script contains the following:

```
#!/bin/sh
AD_TOP=/oracle/OA12/apps/apps_st/appl/ad/12.0.0
export AD_TOP
ADJREOPTS="-ms128m -mx256m -Xrs"
export ADJREOPTS

NLS_LANG=American_America.WE8MSWIN1252
export NLS_LANG

ZX_TOP=/oracle/OA12/apps/apps_st/appl/zx/12.0.0
export ZX_TOP

TEST1=test1
export TEST1
TEST2="test two"
export TEST2

TEST3=" test 3 "
export TEST3
```

Example: Source UNIX Environment Variables from a Script Using EWAGLOBALPROFILE

This example uses the EWAGLOBALPROFILE environment variable to source environment variables from a script. The agent runs as root and the oscomponent.loginshell agent parameter is set to true. Before the echo command runs, the test_var_ewa_global.txt script is sourced and the variables in the file are resolved.

```
AGENT UNIXAGT
USER qatest
CMDNAME /usr/bin/echo
ARGS '$TEST_EWA_GLOBAL > %data/variable_test_EWA_global.txt'
ENVAR EWAGLOBALPROFILE=%var_dir/test_var_ewa_global.txt
```

The test_var_ewa_global.txt script contains the following:

```
#!/bin/ksh
TEST_EWA_GLOBAL="test ewa global"
export TEST_EWA_GLOBAL
```

Example: Set HP Integrity NonStop Environment Variables

In this example, the ENVAR statements do the following:

- Set the full path of an alternative standard output stream to "$C35.PROD.oabcout1" in append mode.
- Set the full path of an alternative standard error output stream to "$c35.PROD.err1" in append mode.
- Set environment variable DEF1 to "DABC;CLASS=MAP;FILE=$c35.PROD.GETDEF".

```
USER prod.glsys
AGENT    PROAGENT
COMMAND  $C35.PROD.GETDEF
ENVAR STDOUT=>>$C35.PROD.oabcout1
ENVAR STDERR=>>$c35.PROD.err1
ENVAR DEF1="DABC;CLASS=MAP;FILE=$c35.PROD.GETDEF"
```

EVENT Statement—Specify the SAP Event to Monitor or Trigger

The EVENT statement specifies the SAP event to monitor or trigger.

Supported Job Type

This statement is required for the SAP Event Monitor job type (see page 162).

Syntax

This statement has the following format:

```
EVENT 'event_name' TRIGGER|REGISTER [PARM(event_parameter)]
     [CONTINUOUS(alertid)]
```

event_name

Specifies the name of the SAP event to monitor or trigger.

Limits: Up to 32 valid SAP characters; case-sensitive

TRIGGER|REGISTER

Specifies the action to take.

TRIGGER

Triggers the SAP event.

REGISTER

Monitors for the SAP event.

PARM(*event_parameter*)

(Optional) Specifies the name of the SAP event parameter.

Limits: Up to 65 valid SAP characters

CONTINUOUS (*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the SAP event is triggered (raised). Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the SAP event is triggered (raised).

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: The CONTINUOUS operand is only applicable to the REGISTER action. The REGISTER action performs the monitoring of SAP events.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Monitor an SAP event

This example monitors the SAP event named SAP_Event. When the SAP event is raised, the job completes.

```
SAPJOBNAME CYBERSAP1
AGENT SAPHTAGENT
EVENT 'SAP_Event' REGISTER PARM(2)
```

Example: Trigger an SAP event

This example triggers the SAP event named SAP_Event. The event parameter is 2.

```
SAPJOBNAME CYBERSAP3
AGENT SAPHTAGENT
EVENT 'SAP_Event' TRIGGER PARM(2)
```

Example: Trigger Alert When the SAP Event is Raised

This example monitors the SAP event named SAP_Event. When the SAP event is raised, the alrt alert is triggered on the scheduling manager.

```
SAPJOBNAME CYBERSAP2
AGENT SAPHTAGENT
EVENT 'SAP_Event' REGISTER CONTINUOUS(alrt)
```

EVENTCATEGORY Statement—Specify a Windows Event Category

The EVENTCATEGORY statement specifies the event category as displayed in the Windows Event Viewer.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTCATEGORY event_category
```

event_category

Specifies the event category as displayed in the Windows Event Viewer.

Limits: Up to 256 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT MONAGT
EVENTLOG Application
EVENTCATEGORY Disk
EVENTID EQ 14
```

EVENTCOMPUTER Statement—Specify a Local Computer for Windows Event Log

The EVENTCOMPUTER statement specifies the name of the local computer that the Windows event log applies to.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTCOMPUTER computer_name
```

computer_name

Specifies the local computer name where the event log is monitored.

Limits: Up to 80 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Monitor a Windows Event Log on a Specific Computer

This example monitors the D9GBJG11 computer for an event source of Security:

```
AGENT WINAGENT
EVENTLOG Security
EVENTTYPE AUDITS
EVENTSOURCE Security
EVENTCOMPUTER D9GBJG11
EVENTCATEGORY 'System Event'
EVENTID GT 0
```

EVENTDESCRIPTION Statement—Specify a Windows Event Description

The EVENTDESCRIPTION statement specifies the description of the event as displayed in the Windows Event Viewer.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTDESCRIPTION event_description
```

event_description

Specifies the event description as displayed in the Windows Event Viewer.

Limits: Up to 256 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 signs. The plus sign is the default and is optional.

```
AGENT WINAGENT
EVENTLOG System
EVENTTYPE Error
EVENTDESCRIPTION '+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.

```
AGENT WINAGENT
EVENTLOG Application
EVENTTYPE info
EVENTDESCRIPTION 'Normal shutdown'
```

EVENTID Statement—Specify the Windows Event IDs to Monitor

The EVENTID statement specifies the Windows event IDs to monitor.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTID operator id
```

operator

Specifies the expression operator. Options are the following:

EQ

Specifies the equal to operator. The job monitors events that have IDs equal to the specified value.

LT

Specifies the less than operator. The job monitors events that have IDs less than the specified value.

LE

Specifies the less than or equal to operator. The job monitors events that have IDs less than or equal to the specified value.

GT

Specifies the greater than operator. The job monitors events that have IDs greater than the specified value.

GE

Specifies the greater than or equal to operator. The job monitors events that have IDs greater than or equal to the specified value.

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.

```
AGENT WINAGENT
LOGNAME Application
EVENTID LT 1
```

EVENTLOG Statement—Specify the Name of a Windows Event Log

The EVENTLOG statement specifies the name of a Windows event log to monitor.

Supported Job Type

This statement is required for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTLOG log_name [CONTINUOUS(alertid)]
```

log_name

Specifies the name of the event log.

Limits: Up to 256 characters; case-sensitive

CONTINUOUS (*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified event log monitoring conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified event log monitoring conditions occur.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: LOGNAME is an alias of EVENTLOG.

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:

```
AGENT WINAGENT
EVENTLOG System
EVENTTYPE WARN
EVENTSOURCE MrxSmb
EVENTCATEGORY None
```

Example: Continuously Monitor an Application Event Log

This example continuously monitors an application event log. An alert named ELOG is triggered for all instances of the INFO event type, where the event source is LLDSAPNT223, the event description contains the word started, and the event ID is less than or equal to 4000.

```
AGENT WINAGENT
EVENTLOG Application CONTINUOUS(ELOG)
EVENTTYPE INFO
EVENTCATEGORY None
EVENTSOURCE LLDSAPNT223
EVENTDESCRIPTION started
EVENTID LE 4000
```

EVENTSOURCE Statement—Specify a Windows Event Source

The EVENTSOURCE statement specifies the Windows event source.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTSOURCE source_name
```

source_name

Specifies the event source as displayed in the Windows Event Viewer.

Limits: Up to 256 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Monitor a Windows Event Source

This example monitors a Windows event source named Service Control Manager. Since this text contains spaces, single quotes are used.

```
AGENT WINAGENT
EVENTLOG System
EVENTTYPE ERROR
EVENTSOURCE 'Service Control Manager'
```

EVENTTIME Statement—Specify the Date and Time of a Windows Event Log

The EVENTTIME statement 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 statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTTIME FROM(time)
```

time

Defines the date and time of the event log. The agent monitors an event log that occurs on or after the specified date and time.

Limits: Up to 80 characters

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Monitor an Application Log that Occurs on or After a Specified Date and Time

This example monitors an application log that occurs on or after a specified date and time. When the job finds an application log that occurs any time on or after 6:30 a.m. on December 11, 2010, the job completes successfully.

```
AGENT WINAGENT
EVENTLOG Application
EVENTTYPE info
EVENTCATEGORY None
EVENTSOURCE LLDSAPNT223
EVENTTIME FROM('12:05:00AM MAR 02 2009')
```

EVENTTYPE Statement—Specify a Windows Event Type

The EVENTTYPE statement specifies the event type to monitor in the Windows event log.

Supported Job Type

This statement is optional for the Windows Event Log Monitoring job type (see page 130).

Syntax

This statement has the following format:

```
EVENTTYPE ERROR|WARN|INFO|AUDITS|AUDITF
```

ERROR

Specifies the Error event type.

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 Information Events

This example monitors the Application log for Information events:

```
AGENT WINAGENT
EVENTLOG Application
EVENTTYPE INFO
EVENTSOURCE Userenv
EVENTCOMPUTER LLUSER
EVENTCATEGORY None
EVENTID EQ 1000
```

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.

```
AGENT WINAGENT
EVENTLOG Security CONTINUOUS(ELOG)
EVENTTYPE AUDITS
EVENTCATEGORY 'System Event'
EVENTDESCRIPTION '-succeeded +Audit log'
```

EXITCODE Statement—Identify Success or Failure by Exit Code

The EXITCODE statement specifies exit codes that indicate the success or failure of a job. If you do not specify the EXITCODE statement, a job is considered to have completed successfully only if a return code of zero (0) is issued.

Supported Job Types

This statement is optional for the following job types:

- i5/OS (see page 113)
- Micro Focus
- UNIX (see page 190)
- Windows (see page 214)

Syntax

This statement has the following format:

```
EXITCODE code [SUCCESS|FAILURE]
```

code

Defines a single exit code or a range of exit codes.

Limits: The maximum length is 41 characters, which allows two 20-digit exit codes plus one dash (-) between them to indicate a range. The minimum exit code value is 0 and the maximum value is 18446744073709551615.

Examples: 4, 10000000000000000000, 0-9999, 10000000000000000000-18446744073709551615

SUCCESS

(Optional) Indicates that the specified exit codes are treated as job success. This is the default.

FAILURE

(Optional) Indicates that the specified exit codes are treated as job failure.

General Notes

Use the EXITCODE statement to indicate an exit code other than 0 or a range of codes as job success.

You can specify multiple exit codes using multiple EXITCODE statements. There is no limit to the number of EXITCODE statements that you can use.

If you specify multiple exit codes, enter the most specific codes first followed by the ranges.

Use the EXITCODE statement within the boundaries of a job definition to apply to a single job or outside the boundaries of the job definition to apply to the entire application.

Example: Define Multiple EXITCODE Statements

In this example, EXITCODE specifies the following conditions:

- Exit codes in the range from 0 to 9999 indicate a successful job completion
- Exit code 19 and exit codes in the range from 10000 to 18446744073709551615 indicate a job failure.

```
EXITCODE 19 FAILURE
EXITCODE 0-9999 SUCCESS
EXITCODE 10000-18446744073709551615 FAILURE
```

EXPIRATION Statement—Specify the Number of Days Before the Spool Request is Deleted

The EXPIRATION statement specifies the number of days before the spool request is deleted.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
EXPIRATION days
```

days

Specifies the number of days before the spool request is deleted. The value corresponds to the SAPGUI Spool options, Spool retention per field on the Background Print Parameters dialog.

Limits: 1-9

Note: If you specify 9 days, the spool request is not deleted.

Notes:

- Use the EXPIRATION statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the EXPIRATION statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Delete Spool Request After Three Days

This example deletes the spool request for the BTCTEST ABAP from the spool system after three days:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   EXPIRATION 3
```

FAILUREMSG Statement—Specify Failure Message for ABAP

The FAILUREMSG statement specifies a string that indicates the failure of the SAP step (ABAP). If the string matches the SAP ABAP output for the step, the step is considered failed.

Supported Job Type

This statement is optional for the SAP R/3 (see page 166) job type.

Syntax

This statement has the following format:

```
FAILUREMSG 'message'
```

message

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

Example: Use a Failure Message to Determine an ABAP's Failure

This example specifies the failure message, 'Internal problem'. If the string is found in the ABAP spool output, the scheduling manager treats the BTCTEST ABAP as failed.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   FAILUREMSG 'Internal problem'
```

FILENAME Statement—Specify File to Monitor

The FILENAME statement specifies the path to and name of a file to monitor. It also specifies the monitor conditions for the file trigger.

Supported Job Type

This statement is required for the File Trigger job type (see page 88).

Syntax

This statement has the following format, except for CA WA Agent for HP Integrity NonStop:

```
FILENAME file_name
{CREATE [SIZE(integer[K|M])]
        [OWNER(owner)][GROUP(group)]
        [NOCHANGE(min)]
        [CONTINUOUS(alertid)]
        [RECURSIVE]}
{DELETE [OWNER(owner)][GROUP(group)]
        [CONTINUOUS(alertid)]
        [RECURSIVE]}
{EXIST [OWNER(owner)][GROUP(group)]
       [RECURSIVE]}
{EXPAND SIZE(integer[K|M])|DELTA(integer[K|M])|PERCENT(integer)
        [OWNER(owner)][GROUP(group)]
        [NOCHANGE(min)]
        [CONTINUOUS(alertid)]
        [RECURSIVE]}
{NOTEXIST [OWNER(owner)][GROUP(group)]
          [RECURSIVE]}
{SHRINK SIZE(integer [K|M])|DELTA(integer [K|M])|PERCENT(integer)
        [OWNER(owner)][GROUP(group)]
        [NOCHANGE(min)]
        [CONTINUOUS(alertid)]
        [RECURSIVE]}
{UPDATE [OWNER(owner)][GROUP(group)]
        [NOCHANGE(min)]
        [CONTINUOUS(alertid)]
        [RECURSIVE]}
```

This statement has the following format for CA WA Agent for HP Integrity NonStop:

```
FILENAME file_name
{CREATE [CONTINUOUS(alertid)]}
{UPDATE [CONTINUOUS(alertid)]}
```

file_name

Specifies the path to and name of the file to monitor.

Limits: Up to 128 characters; case-sensitive

UNIX/Windows: You can use wildcards and spaces in your UNIX and Windows file names and paths.

i5/OS:

- To specify a file in the root file system, use UNIX path and file formats.
- To specify an object in QSYS in path format, use one of the following formats:

  `/QSYS.LIB/`*`library`*`.LIB/`*`object`*`.type/`

  `/QSYS.LIB/`*`library`*`.LIB/`*`object`*`.FILE/`*`member`*`.MBR`
- To specify an object in QSYS in i5/OS standard format, use one of the following formats:

  `library/object/type`

  `library/object/``*FILE(`*`member`*`)`

library

Specifies the name of the library that contains the object. The value must be a valid i5/OS library name.

Limits: Up to 10 characters

Notes:

- You can specify *ALL to match any name.
- You can specify a generic name.

object

Specifies the name of the object. The value must be a valid i5/OS object name.

Limits: Up to 10 characters

Notes:

- You can specify *ALL to match any name.
- You can specify a generic name.

type

Specifies the type of object.

Limits: Up to 10 characters

Note: You can specify *ALL to match any name.

Example: FILE

member

Specifies a member in the *FILE object. The value must be a valid i5/OS member name.

Limits: Up to 10 characters

Notes:

- You can specify *FIRST.
- You can specify *ALL to match any name.

File Trigger Types

Specify one of the following file trigger types to monitor for. For each trigger type, you can specify additional criteria.

CREATE

Indicates that the file trigger occurs when the file is created. This is the default file trigger type.

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. If the file does not exist, the job fails.

Note: If you specify the EXPAND file trigger type, you must specify one of the SIZE, DELTA, or PERCENT operands.

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. If the file does not exist, the job fails.

Note: If you specify the SHRINK file trigger type, you must specify one of the SIZE, DELTA, or PERCENT operands.

UPDATE

Indicates that the file trigger occurs when the file is updated.

Additional Criteria

You can specify the following additional criteria depending on the file trigger type. For information on the criteria that applies to each trigger type, refer to the Syntax.

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified file trigger conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified file trigger conditions occur.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Notes:

- If the file trigger type is CREATE, EXPAND, SHRINK, or UPDATE, the job will not complete unless it is forced to complete.
- If the file trigger type is DELETE, the 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 operand:
  - If the file trigger type is CREATE or DELETE, a wildcard value is not mandatory for the file name value. The following shows how to use wildcards (* or ?): /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 trigger occurs and the agent continues to monitor the files.

    Note: If the CONTINUOUS operand is specified with the NOCHANGE operand and a wildcard is used in the file name value, all of the matching files must remain unchanged for the duration of the NOCHANGE interval. If another file satisfying the trigger criteria is updated during the NOCHANGE interval, the trigger waits until the trigger interval lapses on the first file before monitoring the second file for the duration of the NOCHANGE interval.

DELTA(*integer*[K|M])

Defines the number of bytes the file size must change for the file trigger to occur.

Limits: Up to 10 characters; the first character must be numeric

Note: You can specify the file size in kilobytes (KB) or megabytes (MB) instead of bytes by appending K or M.

GROUP(*group*)

(Optional) Specifies the name of the group that owns the file to be monitored.

Limits: Up to 16 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 file trigger type is CREATE.
  - The job completes if the file trigger type is DELETE or NOTEXIST.
  - The job fails if the file trigger type is EXPAND, EXIST, SHRINK, or UPDATE.

NOCHANGE(*min*)

(Optional) Defines the number of minutes the file must remain unchanged to satisfy the monitor condition.

Limits: Up to 8 numeric digits

Example: 60

OWNER(*owner*)

(Optional) Specifies the user ID that owns the file to be monitored.

Limits: Up to 16 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 user ID, the following occurs:
  - The job continues monitoring if the file trigger type is CREATE.
  - The job completes if the file trigger type is DELETE or NOTEXIST.
  - The job fails if the file trigger type is EXPAND, EXIST, SHRINK, or UPDATE.

PERCENT(*integer*)

Defines the percentage the file size must change for the file trigger to occur.

Limits: Up to 10 characters; the first character must be numeric

RECURSIVE

(Optional) Specifies that the job monitors for file activity in the specified directory and all of its subdirectories.

SIZE(*integer*[K|M])

Defines a limit on the file size in bytes. If the file trigger type is CREATE or EXPAND, the file trigger occurs when the file size is equal to or greater than the specified size. If the file trigger type is SHRINK, the file trigger occurs when the file size is equal to or less than the specified size.

Limits: Up to 10 characters; the first character must be numeric

Note: You can specify the file size in kilobytes (KB) or megabytes (MB) instead of bytes by appending K or M.

i5/OS: When monitoring a *FILE object in QSYS with no member specified in the file name value, 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 default number of seconds the agent waits between scans is 30. If the monitored file changes more than once between scans, the trigger occurs only once, or not at all. For example, if the agent is monitoring updates to a file, and the file is updated twice between scans, the trigger occurs only once. If the agent is monitoring for the creation of a file, and the file is created and deleted between scans, the trigger does not occur. (The file does not exist when the directory is scanned.)
- To change the number of seconds between scans, ask the agent administrator to change the value for the filemonplugin.sleepperiod parameter in the agentparm.txt file. For more information on the filemonplugin.sleepperiod parameter, see the *Implementation Guide* for your agent. For CA WA Agent for HP Integrity NonStop, you cannot change the number of seconds between scans. The value is hard-coded at 60 seconds.
- If the path showing the location of the file contains spaces, put the entire path in single quotation marks, for example:

  ```
  FILENAME 'c:\data files\pay.dat' CREATE
  ```

- 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:

  ```
  FILENAME '/usr/data/*' CREATE
  ```

  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:

  ```
  FILENAME '/usr/data/????.*' CREATE
  ```

- 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 single 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:

  ```
  FILENAME /usr/data/* UPDATE
  ```

  The /* characters are interpreted as the beginning of a comment, so the scheduling manager interprets the statement as follows:

  ```
  FILENAME /usr/data CREATE     /* UPDATE
  ```

  Instead of monitoring for updates to all files in the /usr/data/ directory, the agent monitors for the data file in the /usr/ directory. Also, since CREATE is the default file trigger type and UPDATE is considered part of the comment, the CREATE file trigger type is used instead of the intended UPDATE file trigger type.
- 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.

- 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 statement monitors for the creation of any file object in the QSYS file system with a file name that matches the PAY* generic name:

  ```
  FILENAME 'LIB/PAY*/*FILE' CREATE
  ```

  Note: A name cannot contain only a single asterisk and no other characters. To specify all possible names, use the special value *ALL.

CREATE file trigger type Notes:

- If the file name value contains wildcards, the first matching file is selected.
- If the file exists when the job is readied, the trigger occurs immediately. If the file does not exist when the job is readied, 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 OWNER or GROUP operands are specified, the file trigger occurs only when all specified criteria are satisfied, including the OWNER and GROUP criteria. For example, the following statement monitors for the creation of a file named payroll, owned by JDOE:

  ```
  FILENAME /usr/data/payroll CREATE OWNER(JDOE)
  ```

  If the payroll file is created, but it is not owned by JDOE, the job does not complete. It waits until all specified criteria are satisfied, including the OWNER criteria. If you change the owner of the file to JDOE, the job completes.

DELETE file trigger type Notes:

- If the file does not exist when the job is readied, 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 operand with DELETE only when you are using a wildcard (* or ?) in the file name value. For example, the following statement triggers an alert named INFO when a file whose name begins with pay is deleted from the /usr/data/ directory:

  ```
  FILENAME /usr/data/pay* DELETE CONTINUOUS(INFO)
  ```

EXIST file trigger type Notes:

- If the file does not exist, the job fails.
- If the file name value contains wildcards, the file with the most recent modification time that matches the criteria is monitored.

EXPAND file trigger type Notes:

- If the file does not exist when the file trigger is set, the job fails.
- Deleting the monitored file causes the job to fail.
- If the job monitors for the file size to reach a value of *n* (SIZE(*n*)), the file is monitored until its size increases to *n* or greater than *n*. If the file size at initial monitor time is equal to or greater than *n*, the job completes.
- If the job continuously monitors for the file size to reach a value of *n* (SIZE(*n*) CONTINUOUS), the trigger occurs when the size increases to *n* or more than *n*. If the file size at initial monitor time is equal to or greater than *n*, the trigger occurs immediately. After the trigger occurs, the upper limit is set to the new file size and the job continues monitoring based on the new upper limit. For example, this table shows when the trigger will occur if EXPAND SIZE(10K) is set:

The file size when the Agent scans the Directory	Does the file trigger occur?
15 KB	Yes. The file is bigger than 10 KB when the directory is first scanned. 15 KB becomes the new upper limit.
20 KB	Yes. 20 KB exceeds the upper limit of 15K. 20 KB becomes the new upper limit.
35 KB	Yes. 35 KB becomes the new upper limit.
30 KB	No. The upper limit is now 35K. The size needs to exceed 35 KB for the trigger to occur.
40 KB	Yes. 40 KB exceeds the upper limit of 35K. 40 KB becomes the new upper limit.
45 KB	Yes. 45 KB becomes the new upper limit.
60 KB	Yes. 60 KB becomes the new upper limit.
50 KB	No. The upper limit is now 60K. The size needs to exceed 60 KB for the trigger to occur.
65 KB	Yes. 65 KB exceeds the upper limit of 60K. 65 KB becomes the new upper limit.

- If the job monitors for a change in size by a value of *n* (DELTA(*n*)), the specified file is monitored until it increases by *n*.
- If the job continuously monitors for a change in size by a value of *n* (DELTA(*n*) CONTINUOUS), the trigger occurs when the size increases by *n*. After the trigger occurs, the upper limit is set to the new file size and the job continues monitoring based on the new upper limit.
- If the job monitors for a change in size by percent (PERCENT) and the file size is 0, the trigger occurs when the file size increases.
- If the file name value 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.
- If the OWNER or GROUP operands are specified, the file trigger occurs only when all specified criteria are satisfied, including the OWNER and GROUP criteria. If the OWNER or GROUP does not match, the job fails.

NOTEXIST File Trigger Type Notes:

- 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 value contains wildcards, the file with the most recent modification time that matches the criteria is monitored.

SHRINK file trigger type Notes:

- If the file does not exist when the file trigger is set, the job fails.
- Deleting the monitored file causes the job to fail.
- If the job monitors for the file size to reach a value *n* (SIZE (*n*)), the file is monitored until its size decreases to *n* or less than *n*. If the file size at the initial monitor time is equal to or less than *n*, the job completes.
- If the job continuously monitors for the file size (SIZE(*n*) CONTINUOUS), the trigger occurs when the size decreases to *n* or less than *n*. If the file size at the initial monitor time is equal to or less than *n*, the trigger occurs immediately. After the trigger occurs, the lower limit is set to the new file size and the job continues monitoring based on the new lower limit. If the file reaches 0 bytes, the trigger does not occur and the job continues to run until you manually complete it.
- If the job monitors for a change in file size by a value of *n* (DELTA(*n*)), the specified file is monitored until its size decreases by *n*. If the file size at the time the trigger is set is less than *n*, the job completes when the file reaches size 0. For example, if SHRINK DELTA(10K) is set and the initial file size is 5 kilobytes, the job completes when the file reaches 0 bytes.
- If the job continuously monitors for a change in file size (DELTA(*n*) CONTINUOUS), the trigger occurs when the size decreases by *n*. After the trigger occurs, the lower limit is set to the new file size and the job continues monitoring based on the new lower limit. If the file reaches 0 bytes, the trigger does not occur and the job continues to run until you manually complete it. For example, suppose that the monitored file has an initial size of 25 KB and the following operands are defined:

  ```
  SHRINK DELTA(10K) CONTINUOUS(INFO)
  ```

  - When the file reaches 15 KB, the trigger occurs. The job continues to monitor the file for shrinkage to 5 KB.
  - When the file reaches 5 KB, the trigger occurs. Since the file is less than 10 KB, the job continues to run until you manually complete it.
  - When the file reaches 0 bytes, the trigger does not occur and the job continues to run until you manually complete it.
- If the file name value 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.
- If the OWNER or GROUP operands are specified, the file trigger occurs only when all specified criteria are satisfied, including the OWNER and GROUP criteria. If the OWNER or GROUP criteria does not match, the job fails.

UPDATE file trigger type Notes:

- If the file exists when the job is readied, the trigger occurs when the file is updated.
- If the file does not exist when the job is readied, the job fails.
- If the file exists when the job is readied and is then deleted without modification, the job fails.
- If the OWNER or GROUP operands are specified, the file trigger occurs only when all specified criteria are satisfied, including the OWNER and GROUP criteria. If the OWNER or GROUP criteria does not match, the job fails.
- If the CONTINUOUS operand 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 Increase in File Size to a Certain Amount

This example monitors the test file in the data directory. When the test file reaches one byte or more, the job completes.

```
AGENT NT_NY
FILENAME /data/test EXPAND SIZE(1)
```

Example: Monitor for an Increase in File Size by a Certain Amount

This example monitors the record file in the credit directory. When the record file expands by two megabytes or more, the job completes.

```
AGENT UNIX_TOR
FILENAME /credit/record EXPAND DELTA(2M)
```

Example: Monitor for a Decrease in File Size by a Certain Percentage

This example monitors the test file in the c:\amount directory. When the test file shrinks by 65% or more, the job completes.

```
AGENT NT_TOR
FILENAME c:\amount\test SHRINK PERCENT(65)
```

Example: Check that a File Exists in a Directory

This example monitors the money file in the c:\bank\account directory. If the money file exists in that directory, the job completes. If the money file does not exist in that directory, the job fails.

```
AGENT NT_TOR
FILENAME c:\bank\account\money EXIST
```

Example: Check that a File Exists in a Directory or Its Subdirectories

This example monitors the c:\bank\account directory and its subdirectories for the money file. If the money 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.

```
AGENT NT_TOR
FILENAME c:\bank\account\money EXIST RECURSIVE
```

Example: Check that a File Does Not Exist in a Directory

This example monitors the vacation file in the /start/term/ directory. If the vacation file does not exist in that directory, the job completes. If the vacation file exists in that directory, the job fails.

```
AGENT UNIX_NY
FILENAME /start/term/vacation NOTEXIST
```

Example: Monitor for an Increase in File Size to a Certain Amount with an Unchanged Condition

This example monitors the analysis file in the research directory. When the analysis file expands by 1 byte or more and remains unchanged for 5 minutes or more, the job completes.

```
AGENT UNIX_CHI
FILENAME /research/analysis EXPAND SIZE(1) NOCHANGE(5)
```

Example: Monitor for a Decrease in File Size to a Certain Amount

This example monitors the distribute file in the /cash/items directory. When the distribute file shrinks to 1 KB or less, the job completes.

```
AGENT UNIX_CHI
FILENAME /cash/items/distribute SHRINK SIZE(1K)
```

Example: Monitor for a Decrease in File Size by a Certain Amount

This example monitors the cash file in the c:\cost directory. When the cash file shrinks by 10 bytes, the job completes.

```
AGENT NT_TOR
FILENAME c:\cost\cash SHRINK DELTA(10)
```

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.

```
AGENT UNIX_NY
FILENAME /usr/data/payroll UPDATE RECURSIVE
```

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.

```
AGENT UNIX_NY
FILENAME '/usr/data/*' UPDATE RECURSIVE
```

Example: Use Wildcards to Specify a File Name on the Root File System

This example monitors for files that are created in the /home/cybesp/ directory in the root file system:

```
AGENT I5AGENT
FILENAME '/home/cybesp/PID????.*' CREATE
```

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

Example: Use a Generic Name to Specify a QSYS File Object

In this example, 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 single quotation marks so that the text following /* is not interpreted as a comment.

```
AGENT I5AGENT
FILENAME 'LIB/PAY*/*FILE' CREATE
```

Example: Monitor the Size of a QSYS File Object

In this example, 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.

```
AGENT I5AGENT
FILENAME 'LIB/EXPOBJ/*FILE' EXPAND SIZE(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.

Example: 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 and trigger the same alert. 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.

```
AGENT NT_TOR
FILENAME c:\data\totals EXPAND DELTA(10K) CONTINUOUS(MGMT)

AGENT NT_TOR
FILENAME c:\data\totals SHRINK DELTA(10K) CONTINUOUS(MGMT)
```

Suppose that the initial file size of totals (c:\data\totals) is 100 KB and it changes as follows:

80 KB, 90 KB, 110 KB, 50 KB, 60 KB

The following triggers occur:

- Because of the EXPAND file trigger type, when the file size changes from 90 KB to 110 KB, the first job issues the MGMT alert once.
- Because of the SHRINK file trigger type, the second job issues the MGMT alert 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.

FILESYSTEM Statement—Verify File Space Required to Start a Job

The FILESYSTEM statement 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. Before the job is submitted, the agent checks whether the required space is available on all specified file systems or drives. If all of the requirements are met, the job starts. If any of the requirements are not met, the job fails.

If you do not specify this statement, the available file space is not checked.

Supported Job Types

This statement is optional for the following job types:

- UNIX (see page 190)
- Windows (see page 214)

Syntax

This statement has the following format:

```
FILESYSTEM location SIZE(size [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])
```

location

Specifies a Windows drive or the full path name to a UNIX file system where the file space is required.

Limits: Up to 255 characters

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(*size* [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])

Specifies the required amount of file space in bytes, kilobytes (default), megabytes, or gigabytes.

Limits: Up to 8 digits, followed by the unit of measure

Example: 200 MEGABYTES

Note: To specify multiple file systems or drives, define a separate FILESYSTEM statement for each file system or drive.

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.

```
AGENT UNIXAGENT
SCRIPTNAME /u1/procrun.sh
FILESYSTEM /roots SIZE(100)
FILESYSTEM /auxfsl SIZE(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.

```
AGENT WINAGENT
CMDNAME C:\Programs\Payroll\pay.exe
FILESYSTEM C SIZE(100)
FILESYSTEM D SIZE(120)
```

FILTER Statement—Specify a Filter Using Regular Expression Logic (Application Services/Web Service Jobs)

The FILTER statement specifies a regular expression to use as a filter in a job.

Supported Job Types

This statement is optional for the following job types:

- HTTP (see page 42)
- JMS Subscribe (see page 49)
- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Operation (see page 56)
- Web Service (see page 211)

Syntax

This statement has the following format:

```
FILTER filter
```

filter

Specifies a regular expression to use as a filter. In a JMS Subscribe job, the filter is used to filter messages from the topic or queue. In a JMX-MBean Attribute or Operation job, the filter is used to check against the job's output. If the output matches the success pattern, the job completes; otherwise, it fails. In an HTTP job, the filter is used to filter the output of the invoked servlet. In a Web Service job, the filter is used to determine if the return value is successful.

Limits: Up to 1024 characters; case-sensitive

Example: <StockQuotes><Stock><Symbol>CA.*

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 messages 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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://172.24.0.0:7001
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue CONTINUOUS(a13)
FILTER abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+
TOPIC N
DESTINATION /export/home/user1/outputfile1
JNDIUSER 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

FILTER Statement—Specify a Filter Using Regular Expression Logic (SNMP Jobs)

The FILTER statement specifies a regular expression to use as a filter in an SNMP Subscribe job.

Supported Job Type

This statement is required for the SNMP Subscribe job type (see page 178).

Syntax

This statement has the following format:

```
FILTER filter
```

filter

Specifies a regular expression to use as a filter. The filter is used to filter SNMP trap information.

Limits: Up to 256 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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: Subscribe to the First SNMP Trap the Agent Receives

Suppose that you want the agent to subscribe to the first SNMP trap it receives. In this example, the name of the MIB file is RFC1213-MIB.mib. When the agent receives the first trap, the job completes.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
FILTER .*
```

FINDERMETHOD Statement—Specify a Finder Method

The FINDERMETHOD statement 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 statement is required for the Entity Bean job type (see page 39).

Syntax

This statement has the following format:

```
FINDERMETHOD finder_method
```

finder_method

Specifies the name of the finder method.

Limits: Up to 1024 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE UPDATE
METHOD changephone
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
MODIFYPARAMETER TYPE(String) VALUE(800-555-0199)
```

FINDERPARAMETER Statement—Specify Finder Parameters

The FINDERPARAMETER statement 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 statement is required for the Entity Bean job type (see page 39).

Syntax

This statement has the following format:

```
FINDERPARAMETER {TYPE(type) VALUE(value)}
                {TYPE(type) ARRAY(value,value,...)}
                {URI(uri)}
```

TYPE(*type*)

Specifies the Java class of the parameter.

Limits: Up to 1024 characters; case-sensitive

Examples: String, java.lang.String, Integer

Note: If the package is not specified, java.lang is assumed.

VALUE(*value*)

Specifies the String value for the method parameter.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of string values for the method parameter.

Limits: Up to 1024 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple parameters, use multiple FINDERPARAMETER statements. Order is important. If the method contains three parameters, specify the first parameter in the first statement, the second parameter in the second statement, and the third parameter in the third statement. 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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE UPDATE
METHOD changephone
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
MODIFYPARAMETER TYPE(String) VALUE(800-555-0199)
```

FTPFORMAT Statement—Specify the FTP Format

FTPFORMAT is an alias of TRANSFERCODETYPE (see page 619).

HOST Statement—Specify Database Resource Location

HOST is an alias of DB_URL (see page 308)

INFOPACK Statement—Identify Name of Business Warehouse InfoPackage on SAP System

The INFOPACK statement identifies the name of a Business Warehouse InfoPackage on the SAP system.

Supported Job Type

This statement is required for the SAP Business Warehouse InfoPackage job type (see page 158).

Syntax

This statement has the following format:

```
INFOPACK 'infopackage'
```

infopackage

Specifies the name of the Business Warehouse InfoPackage.

Limits: Up to 30 valid SAP characters; case-sensitive

Example: Run an InfoPackage on the SAP System

This example runs the ZPAK_6C7ZW2JAXS90AK71WD1CYIN InfoPackage on the SAP system:

```
SAPJOBNAME BWIPTEST
AGENT SAPHTAGENT
INFOPACK 'ZPAK_6C7ZW2JAXS90AK71WD1CYIN'
```

INITIAL_CONTEXT Statement—Specify an Initial Context Factory

The INITIAL_CONTEXT statement specifies the initial context factory to use when creating the initial context in a Entity Bean, Session Bean, and JMS job.

Supported Job Types

This statement is required for the following job types:

- Entity Bean (see page 39)
- JMS Publish (see page 47)
- JMS Subscribe (see page 49)
- Session Bean (see page 65)

Syntax

This statement has the following format:

```
INITIAL_CONTEXT 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 1024 characters; case-sensitive

Example: weblogic.jndi.WLInitialContextFactory

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- INITIAL_CONTEXT_FACTORY is an alias of INITIAL_CONTEXT.

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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(a23)
```

INITIAL_CONTEXT_FACTORY Statement—Specify an Initial Context Factory

INITIAL_CONTEXT_FACTORY is an alias of INITIAL_CONTEXT (see page 382).

INTERACTIVE Statement—Specify Whether to Run a Windows Job in Interactive Mode

The INTERACTIVE statement specifies whether to submit a Windows job in interactive mode or in batch mode. 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 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 statement.

Supported Job Type

This statement is optional for the Windows job type (see page 214).

Syntax

This statement has the following format:

```
INTERACTIVE YES|NO
```

YES

Runs the Windows job in interactive mode.

NO

Runs the Windows job in batch mode. This is the default unless the oscomponent.interactive parameter is set to true in the agent's agentparm.txt file.

Example: Run a Windows Job in Interactive Mode

This example runs a Windows job in interactive mode. The job opens the config.txt file in the Windows notepad application on the Windows desktop.

```
AGENT WINAGENT
CMDNAME notepad.exe
ARGS c:\run_info\config.txt
INTERACTIVE YES
```

INVOCATIONTYPE Statement—Specify the HTTP Method Type

The INVOCATIONTYPE statement indicates the HTTP method type, which can be either GET or POST, in an HTTP job.

Supported Job Type

This statement is required for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
INVOCATIONTYPE 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.

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.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://google.com/search
PARAMETER KEYWORD(q) VALUE('ca workload automation')
AUTHORDER (BASIC, DIGEST, NTLM)
FILTER .*AE.*
```

IPADDRESS Statement—Specify the IP Address to Monitor

The IPADDRESS statement specifies an IP address to monitor.

Supported Job Type

This statement is required for the IP Monitoring job type (see page 125).

Syntax

This statement has the following format:

```
IPADDRESS 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 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

Example: Monitor an IP Address for a Stopped Status

This example monitors a device with DNS name, APPARCL. When the device stops running, the job completes.

```
AGENT SYSAGENT
IPADDRESS APPARCL
STATUS STOPPED WAIT
```

IPPORT Statement—Specify the Port Number at the IP Address to Monitor

The IPPORT statement specifies the port number at the IP address to monitor.

Supported Job Type

This statement is optional for the IP Monitoring job type (see page 125).

Syntax

This statement has the following format:

```
IPPORT [ip_port]
```

ip_port

(Optional) Specifies the port number for the IP address being monitored. The agent attempts to connect to this port.

Limits: Up to 5 digits

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 the localhost device at port 5800. When the job runs, it immediately checks if the device is running. If the device is running, the job completes. If the device is not running, the job fails.

```
AGENT SYSAGENT
IPADDRESS localhost
IPPORT 5800
STATUS RUNNING NOW
```

JNDIUSER Statement—Specify a JNDI User ID

The JNDIUSER statement specifies the JNDI user ID in an Entity Bean, Session Bean, and JMS job. This user ID 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 statement is optional for the following job types:

- Entity Bean (see page 39)
- JMS Publish (see page 47)
- JMS Subscribe (see page 49)
- Session Bean (see page 65)

Syntax

This statement has the following format:

```
JNDIUSER userid
```

userid

Specifies the JNDI user ID to be used to gain access to the connection factory.

Limits: Up to 128 characters; case-sensitive; it cannot contain delimiters (such as spaces)

Notes:

- Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.
- JUSER is an alias of JNDIUSER.

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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

JOB_CRITERIA Statement—Specify the Evaluation Criteria for a Return String

The JOB_CRITERIA statement defines a regular expression used to evaluate a string returned by an SQL statement or a stored procedure.

Supported Job Types

This statement is optional for the following job types:

- Database Stored Procedure (see page 71)
- SQL (see page 81)

Syntax

This statement has the following format:

```
JOB_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.

Limits: Up to 256 characters; case-sensitive

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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- This field 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.
- JOBCRIT is an alias of JOB_CRITERIA.

Example: Define Basic Success Criteria

In the following example, if the SQL query returns a PartNo that begins with IDG, the job completes successfully.

```
AGENT CYBDB1
SQL 'SELECT PartNo FROM Inv_List WHERE Stock > 30'
JOB_CRITERIA PartNo=IDG.*
```

Example: Define a More Complex Success Criteria

In the following example, if the SQL query returns an ID that matches the specified regular expression, the job completes successfully.

```
AGENT CYBDB1
SQL 'SELECT ID FROM Proc_List WHERE Orders > 30'
JOB_CRITERIA 'ID=\w{2,4}'
```

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.

JOBCLASS Statement—Assign a Job Class to a Job

The JOBCLASS statement assigns a job class value to a job. When this statement 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 statement is optional for the following job types:

- CPU Monitoring (see page 122)
- Database Monitor (see page 69)
- Database Stored Procedure (see page 71)
- Database Trigger (see page 76)
- Disk Monitoring (see page 123)
- Entity Bean (see page 39)
- File Trigger (see page 88)
- FTP (see page 96)
- HTTP (see page 42)
- i5/OS (see page 113)
- IP Monitoring (see page 125)
- JMS Publish (see page 47)
- JMS Subscribe (see page 49)
- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Create Instance (see page 55)
- JMX-MBean Operation (see page 56)
- JMX-MBean Remove Instance (see page 57)
- JMX-MBean Subscribe (see page 58)
- Micro Focus
- Oracle E-Business Suite Copy Job (see page 136)
- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)
- PeopleSoft (see page 146)
- POJO (see page 60)

- Process Monitoring (see page 126)
- RMI (see page 62)
- Secure Copy (see page 174)
- Secure FTP (see page 175)
- Session Bean (see page 65)
- SNMP Subscribe (see page 178)
- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)
- SQL (see page 81)
- Text File Reading and Monitoring (see page 127)
- UNIX (see page 190)
- Wake on LAN (see page 206)
- Web Service (see page 211)
- Windows (see page 214)
- Windows Event Log Monitoring (see page 130)
- Windows Service Monitoring (see page 132)

Syntax

This statement has the following format:

`JOBCLASS` *`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 128 characters; it cannot contain delimiters (such as spaces)

Notes:

- 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
  ```

- Your agent administrator can set the initiators.afmjobclassmap_*N* parameter to assign a job class automatically based on the combination of verbs and subverbs that appear in an AFM. For more information about the parameter, see the *Implementation Guide* for CA WA Agent for UNIX, Linux, Windows, or i5/OS.
- The JOBCLASS statement overrides the job class specified in the agentparm.txt file of the agent.
- You can define the JOBCLASS statement globally. The JOBCLASS statement in the job definition overrides the global statement in the application.

Example: Specify a Job Class in a UNIX Job

In this example, the UNIX job runs under job class JOBCLASS2.

```
AGENT UNIXAGENT
JOBCLASS JOBCLASS2
SCRIPTNAME /export/home/payroll/data
```

JOBCOPY Statement—Copy an Existing SAP Job

The JOBCOPY statement copies an existing SAP job with all of its attributes.

Supported Job Type

This statement is required for the SAP Job Copy job type (see page 164).

Syntax

This statement has the following format:

```
JOBCOPY JOBCOUNT(count)
     [TARGETJOBNAME(name)]
     [STEPNUM(step)]
```

JOBCOUNT(*count*)

Specifies the ID of the job to be copied.

Limits: Up to eight digits

Example: 00458131

TARGETJOBNAME(*name*)

(Optional) Specifies the name of the target job.

Default: The name of the source job specified in the SAPJOBNAME statement

Limits: Up to 32 valid SAP characters; case-sensitive

STEPNUM(*step*)

(Optional) 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.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Copy an Existing SAP Job

This example defines an SAP Job Copy job that copies the SAPTEST2 job with job count 00458131.

```
AGENT SAPHTAGENT
JOBCOPY JOBCOUNT(00458131)
SAPJOBNAME SAPTEST2
```

JOBCRIT Statement—Specify the Evaluation Criteria for a Return String

JOBCRIT is an alias of JOB_CRITERIA (see page 389).

JOBD Statement—Specify the Job Description

The JOBD statement specifies the job description for the program submitted.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
JOBD description|
     library/description|
     /QSYS.LIB/library.LIB/description.JOBD
```

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.

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.

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.

```
AGENT I5AGENT
CLPNAME MFGDATA
CURLIB PAY1
JOBD CYB1/DFTJOBD
```

JOBNAME Statement—Specify the Job Name (i5/OS Jobs)

The JOBNAME statement specifies a name for an i5/OS job. This is the name displayed in the job queue.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
JOBNAME 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 @

Note: If you do not specify a JOBNAME statement in the job definition, the job name defaults to the name specified in the CLPNAME, COMMAND, or AS400FILE statement.

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.

```
AGENT I5AGENT
CLPNAME PROCPROG
JOBNAME PROG1
```

JOBNAME Statement—Specify Job Name (SAP Jobs)

JOBNAME is an alias of SAPJOBNAME (see page 556).

JOBOBJECT Statement—Associate Windows Job with a Windows Job Object

The JOBOBJECT statement creates a new Windows job object and associates a Windows job with it or associates a Windows job with an existing Windows job object.

Supported Job Type

This statement is optional for the Windows job type (see page 214).

Syntax

This statement has the following format:

```
JOBOBJECT job_object_name [CREATE|ASSIGN]
    [JOBMEMORY(job_memory [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])]
    [PROCESSMEMORY(process_memory [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])]
    [JOBTIME(job_time)]
    [PROCESSTIME(process_time)]
    [PRIORITY(HIGH|ABOVE_NORMAL|NORMAL|BELOW_NORMAL|IDLE)]
    [PROCESSLIMIT(process_limit)]
```

job_object_name

Specifies the name of the new Windows job object or of an existing Windows job object that you want to add this job to.

Limits: Up to 256 characters; case-sensitive

Note: If you are creating a new Windows job object, the name must be unique; there cannot be an existing Windows job object with the same name.

CREATE|ASSIGN

(Optional) Specifies whether to create a new Windows job object or associate the job with an existing Windows object.

CREATE

Creates a new Windows job object and associates the job with it. This is the default.

ASSIGN

Associates the job with an existing Windows job object.

JOBMEMORY(*job_memory* [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])

(Optional) Defines the maximum virtual memory allocated to *all* processes associated with the Windows job object in bytes (default), kilobytes, megabytes, or gigabytes.

Limits: Up to 12 numeric digits, followed by the unit of measure

Examples: 50 MEGABYTES, 500 KILOBYTES

Note: If the total memory used for all processes associated with the Windows job object exceeds this limit, the job that is trying to use memory fails.

PROCESSMEMORY(*process_memory* [BYTES|KILOBYTES|MEGABYTES|GIGABYTES])

(Optional) Defines the maximum virtual memory allocated to *each* process associated with the Windows job object in bytes (default), kilobytes, megabytes, or gigabytes.

Limits: Up to 12 numeric digits, followed by the unit of measure

Examples: 50 MEGABYTES, 500 KILOBYTES

Note: If the memory used for a single process exceeds this limit, the job fails.

JOBTIME(*job_time*)

(Optional) Defines the maximum CPU time in milliseconds allocated to *all* processes associated with the Windows job object.

Limits: Up to 64 numeric digits

Example: 4000000

Note: If the total CPU time for all processes associated with the Windows job object exceeds this limit, all jobs associated with the job object fail.

PROCESSTIME(*process_time*)

(Optional) Defines the maximum CPU time in milliseconds allocated to *each* process associated with the Windows job object.

Limits: Up to 64 numeric digits

Example: 180000

Note: If the CPU time used for a single process exceeds this limit, the job fails.

PRIORITY(HIGH|ABOVE_NORMAL|NORMAL|BELOW_NORMAL|IDLE)

(Optional) Specifies the process priority for *all* processes in the job object as follows:

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.

BELOW_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.

PROCESSLIMIT(*process_limit*)

(Optional) Defines the maximum number of simultaneously active processes allowed in the Windows job object.

Limits: Up to 12 numeric digits

Example: 10

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you create a new Windows job object and you do not specify an operand, that property has an unlimited value.
- If you assign an existing Windows job object and you do not specify a value for an operand, that property keeps the existing value.

Example: Create a New Windows Job Object and Assign a Windows Job to It

The following job definition creates a Windows job object named payjobsobject. Payjobsobject can use up to 40 MB of memory (41943040 bytes) and 1 hour of CPU time (3600000 milliseconds) for all processes it contains. Each process associated with payjobsobject has a higher than normal priority and can use up to 500 KB of memory (512000 bytes) and 3 minutes of CPU time (180000 milliseconds). Payjobsobject can have up to 10 simultaneously active processes.

```
AGENT WINAGENT
CMDNAME myscript.bat
JOBOBJECT payjobsobject CREATE JOBMEMORY(41943040) +
  PROCESSMEMORY(512000) JOBTIME(3600000) PROCESSTIME(180000) +
  PRIORITY(ABOVE_NORMAL) PROCESSLIMIT(10)
```

JOBQ Statement—Specify the Job Queue for a Program

The JOBQ statement specifies the i5/OS job queue for the submitted program.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
JOBQ jobqueue|
     library/jobqueue|
     /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.

jobqueue

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.

Example: Define the Library and Job Queue Names of 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.

```
AGENT I5AGENT
CLPNAME PAYLOAD
CURLIB PAY1
JOBQ /QYS.LIB/QBASE.LIB/JQUEUE.JOBQ
```

JUSER Statement—Specify a JNDI User ID

JUSER is an alias of JNDIUSER (see page 387).

LANGUAGE Statement—Specify the Language the ABAP Uses

The LANGUAGE statement specifies the language the ABAP uses.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
LANGUAGE language
```

language

Specifies an alphabetic character representing a valid language for the SAP system. The value corresponds to the SAPGUI ABAP program Language field on the Create Step dialog.

Default: The SAP system default logon language

Examples: E=English, D=German, R=Russian

Notes:

- Use the LANGUAGE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the LANGUAGE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Language the ABAP Uses

This example uses the English language for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   LANGUAGE E
```

LDA Statement—Specify the Data for the Local Data Area

The LDA statement 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 statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
LDA data|X'hh...'
```

data

Specifies the data in character format.

Limits: Up to 1024 characters; case-sensitive

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'

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify Data for the Local Data Area in Hexadecimal Format

In the following example, when the job is submitted, the agent initializes the local data area with hexadecimal data. When the job completes, the local data area is destroyed automatically by the operating system.

```
AGENT A03TS1
CLPNAME IVP
LDA X'C1C2C3C4'
```

LIBL Statement—Specify the Libraries for an I5/OS Job

The LIBL statement specifies the names of the libraries that the job uses.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
LIBL library|(library 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. The libraries are searched in the same order as they are listed.
- To specify multiple libraries in an LIBL statement, separate each library name with a space and enclose the entire list of libraries in parentheses.
- The LIBL statement 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 LIBL statement overrides the library list specified in the job description for the job.
- You can specify multiple LIBL statements in a job definition.

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.

```
AGENT I5AGENT
CLPNAME PAYJOB
LIBL PAYROLL
```

Example: Add More than One Library to the Library List of an i5/OS Job

In this example, the libraries PAYROLL and FINANCE are added to the library list.

```
AGENT I5AGENT
CLPNAME ACCTJOB
LIBL (PAYROLL FINANCE)
```

LINES Statement—Specify the Number of Lines per List Page

The LINES statement specifies the number of lines per list page.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
LINES num
```

num

Specifies the number of lines per list page. The length of the list is determined by its content. The value corresponds to the SAPGUI Print settings, Report page, Rows field on the Background Print Parameters dialog.

Limits: 0-486

Note: You can use 0 or blank only when viewing the list online. You can not use 0 or blank to format a list to be printed.

Notes:

- Use the LINES statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the LINES statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Print 15 Lines per List Page

This example prints 15 lines per list page for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   LINES 15
```

LOCALFILENAME Statement—Specify Local Filenames (FTP Job)

The LOCALFILENAME statement specifies the name of one or more files on the agent computer involved in an FTP transfer.

Supported Job Types

This statement is required for the following FTP job type (see page 96).

Syntax

This statement has the following format:

```
LOCALFILENAME file_name[; file_name...]
```

file_name

Specifies the destination of the file (if downloading) or the source location of the file (if uploading).

Limits: The total of all files specified cannot exceed 256 characters; case-sensitive

UNIX/Windows:

- If you are downloading a file, specify the full path and file 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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 number of files specified in the LOCALFILENAME and REMOTEFILENAME statements must match.
- LOCALNAME is an alias of LOCALFILENAME.
- LOCALFILENAME can be a global statement.

Example: Download a Single File

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:

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /u1/qatest/ftpdata/textfile
LOCALFILENAME /export/home/ftpfiles/ftpdata/Folder/textfile
```

Example: Download Multiple Files

This example downloads multiple ASCII files from a remote UNIX server. Note that the number of files listed in the REMOTEFILENAME and LOCALFILENAME statements must match. Single quotes enclose both lists to accommodate the line continuation character (+).

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME '/u1/test/scripts/echo;+
  /u1/test/scripts/echo1;+
  /u1/test/scripts/echo2'
LOCALFILENAME '/export/home/test/ftpdata/echo;+
  /export/home/test/ftpdata/echo_1;+
  /export/home/test/ftpdata/echo_2'
```

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.

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION U
TRANSFERCODETYPE A
REMOTEFILENAME /tmp
LOCALFILENAME /export/home/qatest/ftpdata/text*
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5agent
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

LOCALNAME Statement—Specify Local Filenames (FTP Jobs)

LOCALNAME is an alias of LOCALFILENAME (see page 407).

LOCALNAME Statement—Specify the Local File to Transfer (Secure Copy and Secure FTP Jobs)

The LOCALNAME statement specifies a file on the agent computer to be downloaded or uploaded in a secure file transfer.

Supported Job Types

This statement is required for the following job types:

- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
LOCALNAME file_name
```

file_name

Specifies the file's destination (if downloading) or the file's source location (if uploading).

Limits: Up to 256 characters; case-sensitive

Notes:

- For downloads, you must specify the full path and file name without wildcards.
- For uploads, you can use wildcards for the file name if you are using the Secure FTP job. You cannot use wildcards if you are using the Secure Copy job. The asterisk (*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- You cannot rename files if wildcards are used.
- You cannot use wildcards in the path.

Notes:

- Enclose values that contain delimiters (such as spaces) in single quotation marks.
- If a wildcard is used in a local file name for upload, the REMOTENAME statement is not required. A wildcard transfer is equivalent to an mget transfer using an FTP client.

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):

```
AGENT WINAGENT
LOCALNAME C:\temp\install.log1
REMOTENAME install.log
REMOTEDIR /root
SERVERADDR chi-linux
SERVERPORT 22
TRANSFERDIRECTION DOWNLOAD
USER causer
```

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). Since the LOCALNAME statement contains a wildcard, the REMOTENAME statement is not specified.

```
AGENT WINAGENT
SERVERADDR aixunix
TRANSFERDIRECTION UPLOAD
USER causer
REMOTEDIR /u1/build/uploaded
LOCALNAME c:\temp\upload\*
```

LOCALUSER Statement—Specify a User ID on the Agent Computer for the Downloaded File

The LOCALUSER statement specifies a user ID on the computer where the agent is installed. This user ID determines the access permissions of a downloaded file on the agent computer. When the file is downloaded, the file is created with this user as the file owner.

Note: This statement only applies to CA WA Agent for UNIX or Linux. To set the owner of a downloaded file, the agent must run as root.

Supported Job Types

This statement is optional for the following job types:

- FTP (see page 96)
- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
LOCALUSER user_id
```

user_id

Specifies a user ID on the computer where the agent is installed.

Limits: Up to 128 characters; case-sensitive; cannot contain delimiters (such as spaces)

Notes:

- This statement only applies to downloads (TRANSFERDIRECTION set to D or DOWNLOAD).
- The local user ID does not require a password to be stored on the scheduling manager.
- Your agent administrator can specify a default local user for all FTP, Secure Copy, and Secure FTP jobs by setting the ftp.download.owner parameter in the agent's agentparm.txt file.
- The LOCALUSER statement overrides the default setting specified in the ftp.download.owner parameter in the agent's agentparm.txt.

Example: Change the Owner of a File Downloaded from a Remote Server

This example downloads the file_size8 file from the /ul/test/ftpdata directory on the simon server using the Secure Copy Protocol (SCP). After the file is downloaded, the agent computer contains a file named scp_file_size8_local_user in the /u1/causer/data directory. The owner of the file is test. The remote operating system type is UNIX.

```
AGENT UNIXAGENT
LOCALNAME /ul/causer/data/scp_file_size8_local_user
REMOTENAME file_size8
REMOTEDIR /u1/test/ftpdata
SERVERADDR simon
SERVERPORT 22
TRANSFERDIRECTION DOWNLOAD
USER causer
LOCALUSER test
TARGETOSTYPE UNIX
```

LOCATION Statement—Specify a Service Provider URL

The LOCATION statement specifies the URL of the service provider using dotted decimal notation or DNS name in an Entity Bean, Session Bean, and JMS 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 statement is required for the following job types:

- Entity Bean (see page 39)
- JMS Publish (see page 47)
- JMS Subscribe (see page 49)
- Session Bean (see page 65)

Syntax

This statement has the following format:

```
LOCATION 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

  Example: t3://localhost:7001

- 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

  Example: iiop://172.24.0.0:2809

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROVIDER_URL is an alias of LOCATION.

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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
CREATEMETHOD createcustomer
OPERATIONTYPE CREATE
CREATEPARAMETER TYPE(String) VALUE(customerid)
CREATEPARAMETER TYPE(String) VALUE(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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(a23)
```

LOGNAME Statement—Specify Name of Windows Event Log

LOGNAME is an alias of EVENTLOG (see page 351).

MAC Statement—Specify the Media Access Control (MAC) Address

The MAC statement specifies the Media Access Control (MAC) address of the computer that receives the Wake on LAN (WOL) signal. The MAC address is an integral part of the Ethernet card (NIC) of the computer's motherboard.

Supported Job Type

This statement is required for the Wake on LAN job type (see page 206) .

Syntax

This statement has the following formats:

MAC *xx-xx-xx-xx-xx-xx*

xx-xx-xx-xx-xx-xx

Specifies the Media Access Control (MAC) address of the computer that receives the Wake on LAN (WOL) signal. The MAC address consists of six 2-digit hexadecimal values separated by dashes (-). On UNIX, you can obtain the MAC address using ifconfig (listed under HWaddr). On Windows, you can obtain the MAC address using ipconfig /all (listed under Physical Address).

Note: You can also separate the values using colons (:).

Example: Broadcast the WOL Signal to a MAC Address

This example broadcasts the WOL signal to the server identified by the 172.16.0.0 IP address and the 00-11-43-73-38-DC MAC address. After the WOL signal is sent, the job completes immediately without verifying whether the signal worked.

```
AGENT AGENTNME
BROADCAST 172.16.0.0
MAC 00-11-43-73-38-DC
```

MBEAN Statement—Specify the Name of the MBean

The MBEAN statement 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. In a JMX-MBean Subscribe job, you can also indicate continuous monitoring by specifying an alert name to trigger.

Supported Job Types

This statement is required for the following job types:

- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Create Instance (see page 55)
- JMX-MBean Operation (see page 56)
- JMX-MBean Remove Instance (see page 57)
- JMX-MBean Subscribe (see page 58)

Syntax

This statement has the following format:

- For all job types except the JMX-MBean Subscribe job type:

  `MBEAN 'domain_name:key=value[,key=value,...]'`

- For the JMX-MBean Subscribe job type:

  `MBEAN 'domain_name:key=value[,key=value,...]' [CONTINUOUS(alertid)]`

***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 1024 characters; case-sensitive

Example: 'DefaultDomain:type=SimpleDynamic,index=3'

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the JMX-MBean Subscribe job receives an MBean notification. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the agent detects the notification. The agent ignores all notifications that do no match the filter.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
ATTRIBUTE cachesize
```

Example: Set Up Notifications for Change to an MBean Attribute

Suppose that you want to set up continuous monitoring for changes to the cachesize attribute of the MBean named Config. The job filters the notifications the MBean sends by attribute. Each time the cachesize attribute changes, an alert named CHGA is sent.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config' CONTINUOUS(CHGA)
ATTRIBUTESFILTER cachesize
```

MESSAGECLASS Statement—Specify the Java Class of the JMS Message

MESSAGECLASS is an alias of MSGCLASS (see page 429).

METHOD Statement—Specify the Method to be Invoked Remotely

The METHOD statement specifies the method to be invoked remotely in a job.

Supported Job Types

This statement is required for the following job types:

- Entity Bean (see page 39) when the operation type is UPDATE
- POJO (see page 60)
- RMI (see page 62)
- Session Bean (see page 65)

Syntax

This statement has the following format:

```
METHOD method
```

method

Specifies the method to be invoked remotely. 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 a RMI job, it specifies the method of the remote Java class to invoke.

Limits: Up to 1024 characters; case-sensitive

Examples: hello, concat

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- METHODNAME is an alias of METHOD.

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 to 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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN CybEJBTestBean
METHOD reverse
DESTINATION C:\Makapt15
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE(a23)
```

METHODNAME Statement—Specify the Method to be Invoked Remotely

METHODNAME is an alias of METHOD (see page 419).

MIB Statement—Specify a MIB File Name

The MIB statement specifies the name of the MIB (management information base) file in an SNMP job. In an SNMP Subscribe job, you can also indicate continuous monitoring by specifying an alert name to trigger.

Supported Job Types

This statement is required for the following job types:

- SNMP Subscribe (see page 178)
- SNMP Trap Send (see page 179)

This statement is optional for the following job types:

- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Note: In an SNMP Value Set job, the MIB statement is required if the SNMPNODE statement specifies an OID (object identifier) in string (non-numeric) format.

Syntax

This statement has the following format:

- For the SNMP Trap Send, SNMP Value Get, and SNMP Value Set job types:

  `MIB file_name`

- For the SNMP Subscribe job type:

  `MIB file_name [CONTINUOUS(alertid)]`

file_name

Specifies the relative or absolute path to and name of the MIB file.

Limits: Up to 256 characters; case-sensitive

Note: If the MIB file is not loaded successfully, the job fails.

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when an SNMP trap matches the filter condition in an SNMP Subscribe job. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the agent detects the filter value.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the MIB File Name for an SNMP Trap Send Job

Suppose that you want to send the cybtrapstart trap to a network device using SNMP v3. In this example, five string parameters are passed to the trap. The host name of the network device is localhost and its port is 162. The job uses the AES privacy protocol and the SHA authentication protocol. The name of the MIB file is RFC1213-MIB.mib and the default engine ID is used. The credentials of user1 are used for authorization.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE cybtrapstart HOST(localhost) PORT(162) VERSION(3)
SNMPUSER user1
AUTHPROTOCOL SHA
PRIVPROTOCOL AES
ENGINEID
PARAMETER TYPE(snmp:string) VALUE(p1)
PARAMETER TYPE(snmp:string) VALUE(p2)
PARAMETER TYPE(snmp:string) VALUE(p3)
PARAMETER TYPE(snmp:string) VALUE(p4)
PARAMETER TYPE(snmp:string) VALUE(p5)
```

Example: Specify the MIB File Name in an SNMP Subscribe Job

Suppose that you want the agent to subscribe to all SNMP traps it receives. In this example, the name of the MIB file is RFC1213-MIB.mib. Whenever the agent receives a trap, an alert named ALRT is sent.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib' CONTINUOUS(ALRT)
FILTER .*
```

MODIFYPARAMETER Statement—Specify Modify Parameters

The MODIFYPARAMETER statement 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 statement.

Supported Job Type

When the operation type is UPDATE, this statement is required for the Entity Bean job type (see page 39).

Syntax

This statement has the following format:

```
MODIFYPARAMETER {TYPE(type) VALUE(value)}
                {TYPE(type) ARRAY(value,value,...)}
                {URI(uri)}
```

TYPE(*type*)

Specifies the Java class of the parameter.

Limits: Up to 1024 characters; case-sensitive

Examples: String, java.lang.String, Integer

Note: If the package is not specified, java.lang is assumed.

VALUE(*value*)

Specifies the String value for the method parameter.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of string values for the method parameter.

Limits: Up to 1024 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple parameters, use multiple MODIFYPARAMETER statements. Order is important. For example, if the method contains three parameters, specify the first parameter in the first statement, the second parameter in the second statement, and the third parameter in the third statement. 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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE UPDATE
METHOD changephone
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
MODIFYPARAMETER TYPE(String) VALUE(800-555-0199)
```

MON_COND Statement—Specify a Condition to Monitor

The MON_COND statement specifies a condition to monitor the database for.

Supported Job Type

This statement is optional for the Database Monitor job type (see page 69).

Syntax

This statement has the following format:

```
MON_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

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- MONCOND is an alias of MON_COND.

Example: Monitor for Increases in the Number of Rows in a Table Using a Condition

In the following example, the table Inventory_List is monitored continuously for an increase in the number of rows. When the number of rows increases, if the number of units of ProductA has fallen below 100000, the scheduling manager triggers the predefined alert LOW.

```
AGENT CYBDB1
TABLE_NAME Inventory_List CONTINUOUS(LOW)
MON_TYPE I
MON_COND 'ProductA < 100000'
```

MON_TYPE Statement—Specify the Type of Database Change to Monitor

The MON_TYPE statement specifies the type of database change to monitor for.

Supported Job Type

This statement is required for the Database Monitor job type (see page 69).

Syntax

This statement has the following format:

```
MON_TYPE I|D|ID
```

I

Monitors for an increase in the number of rows in the database table.

D

Monitors for a decrease in the number of rows in the database table.

ID

Monitor for an increase or a decrease in the number of rows in the database table.

Notes:

- The specified table is polled (by default, every 10 seconds) for changes to the number of rows.
- MONTYPE is an alias of MON_TYPE.

Example: Monitor for an Increase in the Number of Rows in a Table

In the following example, the table Inventory_List is monitored for an increase in the number of rows in the table. By default, the database is checked for changes every 10 seconds. When the number of rows increases the job completes.

```
AGENT CYBDB1
TABLE_NAME Inventory_List
MON_TYPE I
```

MONCOND Statement—Specify a Condition to Monitor

MONCOND is an alias of MON_COND (see page 425).

MONITORDELAY Statement—Define the Time to Wait Before Checking the Status of Children Programs

The MONITORDELAY statement defines the number of seconds to wait after an Oracle Applications program completes before monitoring its children.

Note: When monitoring children, we recommend that you include this statement in your job definition to ensure the agent receives accurate information about the status of program children.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Copy Job (see page 136)
- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
MONITORDELAY seconds
```

seconds

Defines the number of seconds to wait after an Oracle E-Business Suite program completes before monitoring its children.

Limits: Up to six numeric digits

Example: 60

Notes:

- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.monChildrenDelay parameter in the agent's agentparm.txt file.
- The MONITORDELAY statement overrides the default setting specified in the oa.default.monChildrenDelay parameter in the agent's agentparm.txt.
- The MONITORDELAY statement automatically monitors children, so you do not need to specify CHILDMONITOR Y. Even CHILDMONITOR N is overridden by the MONITORDELAY statement.
- When the MONITORDELAY statement is defined in an Oracle E-Business Suite Request Set job, the setting is applied to all of the programs in the request set. You cannot specify a different setting for each program.

Example: Delay Checking of Children Status in an Oracle E-Business Suite Single Request Job

In this example, the agent checks the status of the single request program's children 60 seconds after the program completes.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
MONITORDELAY 60
```

Example: Delay Checking of Children Status in an Oracle E-Business Suite Copy Job

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

MONTYPE Statement—Specify the Type of Database Change to Monitor

MONTYPE is an alias of MON_TYPE (see page 426).

MSGCLASS Statement—Specify the Java Class of the JMS Message

The MSGCLASS statement specifies the Java class of the JMS message in a JMS Publish job.

Supported Job Type

This statement is required for the JMS Publish job type (see page 47).

Syntax

This statement has the following format:

```
MSGCLASS class
```

class

Specifies the Java class of the JMS message.

Limits: Up to 1024 characters; case-sensitive

Note: If the package is not specified, java.lang is assumed.

Note: MESSAGECLASS is an alias of MSGCLASS.

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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

NOTIFYDUSERS Statement—Specify Users to Notify Using Display Names

The NOTIFYDUSERS statement specifies a list of users to notify when the single request completes using display names.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Single Request job type (see page 142).

Syntax

This statement has the following format:

```
NOTIFYDUSERS (display_user[,display_user...])
```

display_user

Specifies each user by display name.

Limits: case-sensitive

Notes:

- The entire value can be up to 1024 characters.
- Specify multiple user names by separating them with commas or blank spaces.
- To specify a user name with delimiters (such as spaces), enclose the user name in single quotes.

Example: Specify Display User Names

In this example, the Applications Manager and System Administrator users are notified when the single request completes.

```
AGENT OAAGENT
PROGRAM FNDSCURS
DISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
DESC 'AM Test Desc'
ARGS ,,,ALL,,10,
SAVEOUTPUT No
ARGDEFAULTS No
CHILDMONITOR Yes
COPIES 2
MONITORDELAY 5
OUTPUTFORMAT PDF
NOTIFYDUSERS ('Applications Manager', 'System Administrator')
```

NOTIFYUSERS Statement—Specify Users to Notify Using Short Names

The NOTIFYUSERS statement specifies a list of users to notify when the single request completes using short names.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Single Request job type (see page 142).

Syntax

This statement has the following format:

```
NOTIFYUSERS (short_user[,short_user...])
```

short_user

Specifies each user by display name.

Limits: case-sensitive

Notes:

- The entire value can be up to 1024 characters.
- Specify multiple user names by separating them with commas or blank spaces.
- To specify a user name with delimiters (such as spaces), enclose the user name in single quotes.

Example: Specify Short User Names

In this example, the APPSMGR and SYSADMIN users are notified when the single request completes.

```
AGENT OAAGENT
PROGRAM FNDSCURS
DISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
DESC 'AM Test Desc'
ARGS ,,,ALL,,10,
SAVEOUTPUT No
ARGDEFAULTS No
CHILDMONITOR Yes
COPIES 2
MONITORDELAY 5
OUTPUTFORMAT PDF
NOTIFYUSERS (APPSMGR, SYSADMIN)
```

OAUSER Statement—Specify an Oracle Applications User Name

The OAUSER statement specifies an Oracle Applications user name that the job runs under.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Copy Job (see page 136)
- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
OAUSER 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)

Oracle E-Business Suite Single Request or Request Set Job Notes:

- If you do not specify this statement 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 OAUSER statement overrides the default user name specified in the agent's agentparm.txt.

Oracle E-Business Suite Copy 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 OAUSER statement 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 RESPNAME statement 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 a User Name in an Oracle E-Business Suite Single Request Job

In the following example, the user name for the Application Object Library application is SYSADMIN.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

Example: Override the User Name in an Oracle E-Business Suite Copy Job

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

OBJNAME Statement—Identify Name of SAP Archiving Object

OBJNAME is an alias of ARCOBJNAME (see page 239).

OPERATION Statement—Specify the Operation to be Invoked

The OPERATION statement specifies the MBean operation to be invoked in a JMX-MBean Operation job or the web service operation to be invoked in a Web Service job.

Supported Job Types

This statement is required for the following job types:

- JMX-MBean Operation (see page 56)
- Web Service (see page 211)

Syntax

This statement has the following format:

- For the JMX-MBean Operation job type:

  ```
  OPERATION operation
  ```

- For the Web Service job type:

  ```
  OPERATION operation [ONEWAY]
  ```

operation

Specifies the operation to be invoked.

Limits: Up to 1024 characters; case-sensitive

ONEWAY

(Optional) Invokes the service one way. After the agent invokes the operation, the Web Service job completes without waiting for a response.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
OPERATION resetmem
PARAMETER TYPE(Integer) VALUE(50)
```

OPERATIONTYPE Statement—Specify the Operation Type

The OPERATIONTYPE statement 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 statement is required for the Entity Bean job type (see page 39).

Syntax

This statement has the following format:

```
OPERATIONTYPE 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 FINDERMETHOD, FINDERPARAMETER, METHOD, and MODIFYPARAMETER statements.
- To remove an entity bean, you require the FINDERMETHOD and FINDERPARAMETER statements.

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.

```
AGENT APPAGENT
INITIAL_CONTEXT weblogic.jndi.WLInitialContextFactory
LOCATION t3://localhost:7001
BEAN customer
OPERATIONTYPE REMOVE
FINDERMETHOD acme
FINDERPARAMETER TYPE(String) VALUE(customerid)
```

OTHERS Statement—Pass Keyword Parameters to SBMJOB

When the scheduling manager submits a job to the i5/OS system, the job must pass through the SBMJOB command to execute. Keyword parameters are additional parameters that the scheduling manager passes to the i5/OS SBMJOB command. The OTHERS statement lets you specify any valid SBMJOB command keyword and value combination. You can also specify multiple combinations.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

`OTHERS` *`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).

Notes:

- You can specify multiple keyword and value combinations. Separate each combination with a space.
- The entire value can be up to 4078 characters.
- The scheduling manager does not validate the keyword and value combinations.
- The agent uses the user's default output queue if the USER statement is specified in the job definition and the OTHERS statement does not include the SBMJOB OUTQ( ) command keyword and value that defines the output queue for the job.

Example: Define 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.

```
AGENT I5AGENT
CLPNAME PAYJOB
OTHERS PRTDEV(*JOBD) OUTQ(*JOBD)
```

OUTDESTFORMAT Statement—Specify the Output Format for a PeopleSoft Report

The OUTDESTFORMAT statement specifies the output format for the PeopleSoft report.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
OUTDESTFORMAT 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
0	Any	Any
1	None	(None)
2	PDF	Acrobat (*.pdf)
3	CSV	Comma delimited (*.csv)
4	HP	HP Format (*.lis)
5	HTM	HTML Documents (*.htm)
6	LP	Line printer format (*.lis)
7	WKS	Lotus 1-2-3 files
8	XLS	Microsoft Excel Files (*.xls)
9	DOC	Microsoft Word (*.doc)
10	PS	Postscript (*.lis)
11	RPT	Crystal Report (*.rpt)
12	RTF	Rich Text File (*.rtf)
13	SPF	SQR Portable Format (*.spf)
14	TXT	Text Files (*.txt)
15	OTHER	Other
16	Default	Default
17	XML	XML Format (*.xml)
18	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 OUTDESTFORMAT statement overrides the default setting specified in the ps.default.outDestFormat parameter in the agent's agentparm.txt.

Example: Specify the Output Format

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.

```
AGENT PS_NY
OUTDESTTYPE FILE
OUTDESTFORMAT TXT
PROCESSNAME PAYROLL
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

OUTDESTPATH Statement—Specify an Output Path for a PeopleSoft Job

The OUTDESTPATH specifies the output destination for the PeopleSoft request. The destination can be a file directory or a printer. The value you specify for this statement overrides the default destination defined on the PeopleSoft system.

Note: To use this statement, you must specify FILE or PRINTER in the OUTDESTTYPE statement.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

`OUTDESTPATH` *`file_path|printer_path`*

file_path

Specifies the path to the output directory and the output file name.

Limits: Up to 128 characters; case-sensitive

Note: Specify this path when the OUTDESTTYPE statement is set to FILE.

Default: The PeopleSoft log/output directory

printer_path

Specifies the network location of the printer including the printer server and shared printer name.

Limits: Up to 128 characters; case-sensitive

Default: lpt1

Notes:

- Specify this path when the OUTDESTTYPE statement is set to PRINTER.
- The specified printer must be set up on the PeopleSoft system.
- Your agent administrator can specify a default printer by setting the ps.default.printer parameter in the agent's agentparm.txt file.
- This value overrides the default printer specified in the ps.default.printer parameter in the agent's agentparm.txt file.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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.

```
AGENT PS_NY
PROCESSNAME XRFWIN
PROCESSTYPE 'SQR Report'
OUTDESTTYPE FILE
OUTDESTFORMAT PDF
OUTDESTPATH '/export/home/PSoutput/report1.pdf'
RUNCONTROLID test
PSOPRID VP1
```

Note: For a Windows NT example, substitute a Windows path in the OUTDESTPATH statement.

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.

```
AGENT PS_NY
PROCESSNAME XRFWIN
PROCESSTYPE 'SQR Report'
OUTDESTTYPE PRINTER
OUTDESTFORMAT PS
OUTDESTPATH '\\CA\PRINTER1'
RUNCONTROLID test
PSOPRID VP1
```

OUTDESTTYPE Statement—Specify the Output Type for a PeopleSoft Report

The OUTDESTTYPE statement specifies the output destination type for a PeopleSoft report.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
OUTDESTTYPE output_type
```

output_type

Specifies the field number or field name of the output destination type. PeopleSoft stores the list of output destination types in the PSXLATITEM table.

Limits: Up to 8 characters; case-sensitive; it cannot contain delimiters (such as spaces)

Default: NONE

Options are the following:

Field Number	Field Name (short form)
0	ANY (A)
1	NONE (N)
2	FILE (F)

Field Number	Field Name (short form)
3	PRINTER (P)
4	WINDOW (WI)
5	EMAIL (E)
6	WEB (WE)
7	DEFAULT (D)

Notes:

- 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 OUTDESTTYPE statement overrides the default setting specified in the ps.default.outDestType parameter in the agent's agentparm.txt.
- If you specify EMAIL or WEB as the output destination type, you can email the PeopleSoft report to recipients on a distribution list.
- This statement corresponds to the Type field in PeopleSoft.

Example: Specify Email as the Output Type

This example specifies an output destination type of EMAIL with the file format of PDF:

```
AGENT PSAGENT
PROCESSNAME XRFWIN
PROCESSTYPE Crystal
OUTDESTTYPE EMAIL
OUTDESTFORMAT PDF
PSOPRID VP2
```

Example: Specify Printer as the Output Type

This example specifies an output destination type of PRINTER with the file format of PS:

```
AGENT PSAGENT
PROCESSTYPE 'SQR Report'
PROCESSNAME XRFWIN
OUTDESTFORMAT PS
OUTDESTTYPE PRINTER
OUTDESTPATH \\CA\PRINTER1
```

OUTFILE Statement—Specify File for Results of SQL Query

OUTFILE is an alias of OUTPUT_FILE (see page 443).

OUTPUT_FILE Statement—Specify File for Results of SQL Query

The OUTPUT_FILE statement specifies the location of the output file that stores the SQL query results. If you omit this statement, the results are stored in the job's spool file.

Supported Job Type

This statement is optional for the SQL job type (see page 81).

Syntax

This statement has the following format:

```
OUTPUT_FILE outfile
```

outfile

Specifies the location of the output file that stores the SQL query results.

Limits: Up to 256 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default destination file for all SQL jobs by setting the spooldir parameter in the agent's agentparm.txt file.
- The OUTPUT_FILE statement overrides the default file specified in the agent's agentparm.txt.
- OUTFILE is an alias of OUTPUT_FILE.

Example: Store SQL Query Results in a File

In the following example, the output from the SQL statement is directed to file /var/dblog/orderlog.

```
AGENT CYBDB1
SQL 'SELECT * from NEWORDS'
OUTPUT_FILE /var/dblog/orderlog
```

OUTPUTFORMAT Statement—Specify Output Format for a Single Request or Request Set

The OUTPUTFORMAT statement specifies the output format for a single request or request set.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
OUTPUTFORMAT output_format
```

output_format

Specifies the output format for a single request or request set. In Oracle Applications, the output format is specified as a request definition option and is found in the Format column of the Upon Completion dialog.

Limits: Up to ten characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the Output Format

Suppose that you want to define a Single Request job to run a single request program on the CYBOA agent. The single request belongs to the FND application in Oracle Applications and runs the program FNDSCURS. The job runs under the SYSADMIN user with System Administrator responsibility and includes layout template options.

```
AGENT CYBOA
PROGRAM FNDSCURS
APPLSHORTNAME FND
RESPNAME 'System Administrator'
OAUSER SYSADMIN
OUTPUTFORMAT PDF
TEMPLATELANG EN
TEMPLATETERR US
```

PARAM Statement—Specify Positional Parameters (i5/OS Jobs)

The PARAM statement defines the parameter values that you want to pass to the program at the time the program is invoked.

Supported Job Type

This statement is optional for the i5/OS job type (see page 113).

Syntax

This statement has the following format:

```
PARAM value|'value' 'value'...
```

value

Specifies the value of a parameter to pass to the program.

Notes:

- The entire value can be up to 4078 characters.
- To specify multiple values with a single PARAM statement, enclose each value with single quotation marks and separate each value with a space.
- You can specify multiple PARAM statements in a job definition.

Example: Pass Multiple Parameters to an i5/OS Program

This example passes six parameters to an i5/OS program named PAYJOB:

```
AGENT I5AGENT
CLPNAME PAYJOB
PARAM 'ABC' '1' 'P' 'VALUE C' 'X' 'r'
```

PARAM Statement—Specify Data Selection Criteria (SAP Jobs)

The PARAM statement specifies the Business Warehouse InfoPackage data selection criteria.

Supported Job Type

This statement is optional for the SAP Business Warehouse InfoPackage job type (see page 158).

Syntax

This statement has the following format:

```
PARAM FieldName, {ObjectName|""}, {Sign|""}, {Operation|""}, {LowValue|""},
{HighValue|""}
```

FieldName

Specifies the Business Warehouse InfoPackage Technical Name.

Limits: Up to 30 valid SAP characters; case-sensitive

ObjectName

Specifies the Business Warehouse InfoPackage InfoObject.

Limits: Up to 30 valid SAP characters; case-sensitive

Note: If no value is assigned to *ObjectName*, use two double quotation marks ("") as a place holder.

Sign

Specifies the Business Warehouse InfoPackage Sign. Options are the following:

- I—Specifies include.
- E—Specifies exclude.

Note: If no value is assigned to *Sign*, use two double quotation marks ("") as a place holder.

Operation

Specifies the Business Warehouse InfoPackage Operation. Options are the following:

- BT—Specifies between.
- E—Specifies equal.

Note: If no value is assigned to *Operation*, use two double quotation marks ("") as a place holder.

LowValue

Specifies the Business Warehouse InfoPackage From Value.

Note: If no value is assigned to *LowValue*, use two double quotation marks ("") as a place holder.

HighValue

Specifies the Business Warehouse InfoPackage To Value.

Note: If no value is assigned to *HighValue*, use two double quotation marks ("") as a place holder.

Notes:

- To specify multiple data selection criteria, define a PARAM statement for each set of criteria. Each set of criteria represents a row in the SAP table.
- The value of each statement can be up to 4070 characters.
- The fields on the PARAM statement correspond to the fields on the SAP Administrator Workbench RSA1 transaction.

Example: Specify Multiple Data Selection Criteria

In this example, the data selection criteria for the *FieldNames* (SAP InfoObjects) */BIC/IO_CMAT*, *CALDAY*, */BIC/ IO_CCUST*, and */BIC/IO_CSREP* are defined for InfoPackage ZPAK_42RRVI7G0PA005H00BV0ZJV9I on the SAP system.

For the /BIC/IO_CMAT and CALDAY *FieldNames*, values are provided for each operand.

For the /BIC/IO_CCUST *FieldName*, double quotation marks are used as place markers for *ObjectName*, *Sign*, and *Operation*.

For the /BIC/IO_CSREP *FieldName*, double quotation marks are used as place markers for *ObjectName*, *Sign*, *Operation*, *LowValue*, and *HighValue*.

```
AGENT SAPHTNTAGENT
SAPJOBNAME IP05
INFOPACK 'ZPAK_42RRVI7G0PA005H00BV0ZJV9I'
RFCDEST bw2
PARAM /BIC/IO_CMAT, IO_CMAT, I, BT, MAT001, MAT002
PARAM CALDAY, 0CALDAY, I, BT, 20010101, 20050101
PARAM /BIC/IO_CCUST, "", "", "", CUST001,CUST003
PARAM /BIC/IO_CSREP, "", "", "", "", ""
```

PARAMETER Statement—Specify Input Parameters (Application Services Jobs)

The PARAMETER statement specifies input parameters in an Application Services job.

Supported Job Types

This statement is required for the JMX-MBean Attribute Set job type (see page 54).

This statement is optional for the following job types:

- HTTP (see page 42)
- JMS Publish (see page 47)
- JMX-MBean Create Instance (see page 55)
- JMX-MBean Operation (see page 56)
- POJO (see page 60)
- RMI (see page 62)
- Session Bean (see page 65)

Syntax

This statement has the following format:

- For all job types except the HTTP job type:

```
PARAMETER {TYPE(type) VALUE(value)}
          {TYPE(type) ARRAY(value,value,...)}
          {URI(uri)}
```

- For the HTTP job type:

```
PARAMETER KEYWORD(variable) VALUE(value)
```

TYPE(*type*)

Specifies the Java class of the parameter.

Limits: Up to 1024 characters; case-sensitive

Examples: String, java.lang.String, Integer

Note: If the package is not specified, java.lang is assumed.

KEYWORD(*variable*)

Specifies the variable that the agent passes to the program the job invokes.

Limits: Up to 1024 characters; case-sensitive

VALUE(*value*)

Specifies the parameter value. In most job types, it specifies the String value of the parameter. In an HTTP job, it specifies the value for the variable.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of string values for the method parameter.

Limits: Up to 1024 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple parameters, use multiple PARAMETER statements. Order is important. For example, if the method contains three parameters, specify the first parameter in the first statement, the second parameter in the second statement, and the third parameter in the third statement. 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: Specify Input Parameters 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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

Example: Specify Input Parameters in a Session Bean Job

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.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
BEAN Shoppingcart
CREATEMETHOD createaddbook
METHOD addbook
CREATEPARAMETER TYPE(String) VALUE(ISBN)
PARAMETER TYPE(Integer) ARRAY(1551929120,1582701709)
```

Example: Specify Input Parameters in a JMX-MBean Attribute Set Job

Suppose that you want to set the value of the State attribute of the cdc.jmx.SimpleDynamic MBean to the serialized Java object located in a path on the agent computer.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
MBEAN 'DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic'
ATTRIBUTE State
PARAMETER URI('FS:C:\Program Files\CA\WA Agent +
R11.3_B129\spool\CM_QABC\MAIN\LLR113.7\JMXATT.GET.12587292392520')
```

Example: Specify Input Parameters 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.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://google.com/search
PARAMETER KEYWORD(q) VALUE('ca workload automation')
AUTHORDER (BASIC, DIGEST, NTLM)
FILTER .*AE.*
```

PARAMETER Statement—Specify an Input Parameter (HP Integrity NonStop Jobs)

The PARAMETER statement enables you to specify a parameter to pass to an HP Integrity NonStop job. The parameter persists only for the current job and it is set in the environment of the command that the agent runs.

Supported Job Types

This statement is optional for the HP Integrity NonStop job type (see page 105).

Syntax

This statement has the following format:

```
PARAMETER NAME(name) VALUE(value)
```

name

Specifies the parameter name.

Limits: Maximum of 31 characters. Case-sensitive. You can use the following characters: alphanumeric, hyphen, and circumflex (up caret).

Note: Although the name field is case-sensitive, the value of the name when comparing parameters names is not case-sensitive. If duplicate names are detected, the last one encountered is passed to the job. For example, PARAMETER NAME(abc) VALUE(not used) is coded followed by PARAMETER NAME(ABC) VALUE(used), the executing program receives ABC=used.

value

Specifies the parameter value.

Limits: Maximum of 255 characters. Case-sensitive. You can use the following delimiters: comma, semicolon, and space. Enclose the full text string in single quotes when delimiters are in use.

Note: The combined length of all the PARAMETER statements in a job cannot exceed 511 characters minus the number of entities coded. For example, if 64 PARAMETER statements are coded, the maximum length for all PARAMETER data is 511 – 64.

Example: PARAMETER Statement

In this example, the PARAMETER statements set parameter ABC to ‘100’ and parameter XYZ to ‘200’.

```
AGENT PROAGENT
USER prod.glsys
COMMAND $C35.TCSOBJ.OABC
ENVAR STDOUT=$C35.TCSOBJ.cafout1
ENVAR STDERR=$c35.srkobj.caferr1
ASSIGN "$VOL.SUBVOL.FILE1;$VOL1.SUBVOL1.FILE2"
PARAMETER NAME(ABC) VALUE(100)
PARAMETER NAME(XYZ) VALUE(200)
DEFINE "MYFIL1;CLASS=MAP;FILE=$DATA2.TCSDEMO.FINFOJOB"
```

PARAMETER Statement—Specify Input Parameters (SNMP Jobs)

The PARAMETER statement specifies input parameters in an SNMP job.

Supported Job Types

This statement is required for the SNMP Value Set job type (see page 183).

This statement is optional for the SNMP Trap Send job type (see page 179).

Syntax

This statement has the following format:

```
PARAMETER {TYPE(snmp:type) VALUE(value)}
          {TYPE(snmp:type) ARRAY(value,value,...)}
          {[TYPE(snmp:type)] URI(uri)}
```

TYPE(snmp:*type*)

Specifies the SNMP parameter type. Options for the *type* include the following:

int

Specifies an integer value.

uint

Specifies an unsigned integer value.

ticks

Specifies a time ticks (measured in hundreds of a second) value.

addr

Specifies an IP address value.

Note: SNMP only supports IPv4 addresses. IPv6 addresses are not supported.

oid

Specifies an OID (object identifier) value.

string

Specifies an octet string value.

string.x

Specifies an octet string value represented in hexadecimal format.

counter32

Specifies a counter32 value. A count32 value is a non-negative integer that monotonically increases until it reaches a maximum value of 232 - 1 (4294967295). It then wraps around and starts increasing again from zero.

gauge32

Specifies a gauge32 value. A guage32 value is a non-negative integer that can increase or decrease, but cannot exceed a maximum value. The maximum value cannot exceed 232 - 1 (4294967295). The value of a Gauge has its maximum value whenever the information being modeled is greater or equal to that maximum value; if the information being modeled then decreases below the maximum value, the Gauge also decreases.

Limits: Up to 1024 characters; case-sensitive

VALUE(*value*)

Specifies the SNMP parameter value. The value is converted to the corresponding *type*. In an SNMP Trap Send job, it specifies the value of the parameter that you are passing to the trap. In an SNMP Value Set job, it specifies the new value of the variable that you are changing.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of string values for the SNMP parameter. The values are converted to the corresponding SNMP parameter type.

Limits: Up to 256 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple SNMP parameters, use multiple PARAMETER statements. Order is important. The agent passes the parameters in the order that you specify them.

Example: Specify Input Parameters in an Trap Send Job

Suppose that you want to send the cybtrapstart trap to a network device using SNMP v3. In this example, five string parameters are passed to the trap. The host name of the network device is localhost and its port is 162. The job uses the AES privacy protocol and the SHA authentication protocol. The name of the MIB file is RFC1213-MIB.mib and the default engine ID is used. The credentials of user1 are used for authorization.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE cybtrapstart HOST(localhost) PORT(162) VERSION(3)
SNMPUSER user1
AUTHPROTOCOL SHA
PRIVPROTOCOL AES
ENGINEID
PARAMETER TYPE(snmp:string) VALUE(p1)
PARAMETER TYPE(snmp:string) VALUE(p2)
PARAMETER TYPE(snmp:string) VALUE(p3)
PARAMETER TYPE(snmp:string) VALUE(p4)
PARAMETER TYPE(snmp:string) VALUE(p5)
```

Example: Specify Input Parameter in an SNMP Value Set Job

Suppose that you want to set the value of the agentLogLevel variable to its highest level to diagnose a problem. In this example, the host name of the network device is host.example.com and its port is 161. The SNMP version is v2 and the read community is public. The name of the MIB file is RFC1213-MIB.mib, which is located on the agent computer.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE agentLogLevel HOST(host.example.com) PORT(161) VERSION(2)
COMMUNITY public
PARAMETER TYPE(snmp:int) VALUE(8)
```

PARAMETER Statement—Specify Input Parameters (Web Service Jobs)

The PARAMETER statement specifies input parameters in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
PARAMETER {TYPE(xsd:type) VALUE(value)}
          {TYPE(xsd:type) ARRAY(value,value,...)}
          {URI(uri)}
```

TYPE(xsd:*type*)

Specifies the XML schema type of 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

VALUE(*value*)

Specifies the value for the operation parameter.

Limits: Up to 1024 characters; case-sensitive

ARRAY(*value*,*value*,...)

Specifies a set of values for the operation parameter.

Limits: Up to 1024 characters per array value; case-sensitive

Note: If you specify the ARRAY operand, the statement value cannot exceed 3588 characters.

URI(*uri*)

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized Java object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.

Limits: Up to 255 characters; case-sensitive

Example: 'FS:c:\Program Files\Common Files\System\ADO\input.txt'

Note: For more information about retrieving the URI of a payload producing job, see the documentation for your scheduling manager.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple operation parameters, use multiple PARAMETER statements. 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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

PORTNAME Statement—Specify the Port Name Within the Namespace

The PORTNAME statement specifies the WSDL port name within the target namespace in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
PORTNAME 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

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- In a Web Service job, if you specify the WSDL_URL statement but not the ENDPOINT_URL statement, you must specify both the SERVICENAME and PORTNAME statements.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

PRINTCOPIES Statement—Specify the Number of Copies to Print (Oracle E-Business Suite Jobs)

The PRINTCOPIES statement specifies the number of copies to print for an Oracle Applications single request or request set.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
PRINTCOPIES 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 six numeric digits

Example: 100

Notes:

- Your agent administrator can specify a default number of copies for all Oracle E-Business Suite Request Set or Single Request jobs by setting the oa.default.printCopies parameter in the agent's agentparm.txt file.
- The PRINTCOPIES statement overrides the default setting specified in the oa.default.printCopies parameter in the agent's agentparm.txt.
- 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 PROGPRINTCOPIES statement. This statement overrides the PRINTCOPIES statement for that particular program.
- COPIES is an alias of PRINTCOPIES.

Example: Print Three Copies of a Single Request

In this example, the job specifies three copies to be printed on the Q_DEVELOPMENT printer in PORTRAIT style.

```
AGENT CYBOA
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PROGRAM XXCOFI
PRINTER Q_DEVELOPMENT
PRINTSTYLE PORTRAIT
PRINTCOPIES 3
```

PRINTCOPIES Statement—Specify the Number of Print Copies (SAP Jobs)

The PRINTCOPIES statement specifies the number of print copies for the reports associated with the ABAP specified in the job.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTCOPIES num
```

num

Specifies the number of print copies for the reports associated with the ABAP specified in the job. The value corresponds to the SAPGUI Number of copies field on the Background Print Parameters dialog.

Limits: 1-255

Notes:

- Use the PRINTCOPIES statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTCOPIES statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- COPIES is an alias of PRINTCOPIES.

Example: Specify Number of Print Copies

This example prints three copies of each report associated with the ABAP BTCTEST:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   PRINTCOPIES 3
```

PRINTCOVER Statement—Specify the SAP Cover Page Text

The PRINTCOVER statement specifies the text on an SAP cover page.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTCOVER cover_text
```

cover_text

Specifies the cover page text.

Limits: Up to 68 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTCOVER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTCOVER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify Cover Page Text

This example specifies Cover page text as the cover page text for the RSPARAM ABAP. The cover page text requires single quotes because of the spaces.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTCOPIES 2
   PRINTCOVER 'Cover page text'
```

PRINTDATASET Statement—Specify Data Set Name for SAP Print Spool

The PRINTDATASET statement specifies the data set name for the SAP print spool.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTDATASET data_set
```

data_set

Specifies the print spool data set name.

Limits: Up to 6 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTDATASET statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTDATASET statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Print Spool Data Set Name

This example specifies LIST1S as the print spool data set name for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   LINES 65
   COLUMNS 80
   PRINTDATASET LIST1S
   PRINTCOPIES 1
```

PRINTDEPARTMENT Statement—Specify Department Name

The PRINTDEPARTMENT statement specifies the department name to print on the SAP cover page.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTDEPARTMENT department
```

department

Specifies the department name to print on the cover page.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTDEPARTMENT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTDEPARTMENT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify Department Name

This example prints department Department1 on the cover page for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTDEPARTMENT Department1
```

PRINTDEST Statement—Specify Print Destination

The PRINTDEST statement specifies the print destination for an ABAP.

Supported Job Types

This statement is required for the SAP Data Archiving job type (see page 160).

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTDEST destination
```

destination

Specifies the output device name. The value corresponds to the SAPGUI Output device field on the Background Print Parameters dialog.

Limits: Up to 4 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTDEST statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTDEST statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Print Destination

This example specifies dev1 as the print destination for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   PRINTDEST dev1
   LINES 65
   COLUMNS 60
```

PRINTER Statement—Specify a Printer

The PRINTER statement specifies the name of a printer to be used by Oracle Applications.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

`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

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default printer for all Oracle E-Business Suite Request Set or Single Request jobs by setting the oa.default.printer parameter in the agent's agentparm.txt file.
- The PRINTER statement overrides the default setting specified in the oa.default.printer parameter 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 PROGPRINTER statement. This statement overrides the PRINTER statement for that particular program.

Example: Specify a Printer for Oracle Applications

In this example, the job specifies three copies to be printed on the Q_DEVELOPMENT printer in PORTRAIT style.

```
AGENT CYBOA
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PROGRAM XXCOFI
PRINTER Q_DEVELOPMENT
PRINTSTYLE PORTRAIT
PRINTCOPIES 3
```

PRINTFOOTER Statement—Specify Whether to Print a Footer on an SAP Report

The PRINTFOOTER statement specifies whether to print a footer on a SAP report.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTFOOTER Y|N
```

Y

Prints the footer.

N

Does not print the footer.

Notes:

- Use the PRINTFOOTER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTFOOTER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Print Footer on SAP Report

This example prints the footer on the SAP report for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTCOPIES 2
   PRINTFOOTER Y
```

PRINTFORMAT Statement—Specify an Output Format for an SAP Report

The PRINTFORMAT statement specifies an output format for an SAP report.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTFORMAT format
```

format

Specifies an output format.

Limits: Up to 16 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTFORMAT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTFORMAT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify Output Format

In this example, the SAP report uses the output format X_65_80 for the RSPARAM ABAP.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTFORMAT X_65_80
```

PRINTHOSTPAGE Statement—Print Host Page

The PRINTHOSTPAGE statement specifies whether to print a host page.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTHOSTPAGE Y|N|D
```

Y

Prints the host page.

N

Does not print the host page.

D

Uses SAP default.

Notes:

- Use the PRINTHOSTPAGE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTHOSTPAGE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Print Host Page

This example prints the host page for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTHOSTPAGE Y
```

PRINTIMMED Statement—Specify Immediate Print After Job Completion

The PRINTIMMED statement specifies whether to print the job immediately after the job completes.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTIMMED YES|NO
```

YES

Prints the job immediately after completion.

NO

Does not print the job immediately after completion. The job output remains on spool.

Notes:

- Use the PRINTIMMED statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTIMMED statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- PRINTIMMED corresponds to the SAPGUI Spool options, Print immediately field on the Background Print Parameters dialog.

Example: Print the Job Immediately After Completion

This example prints the job immediately after completion for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   PRINTIMMED YES
```

PRINTNEWSPOOL Statement—Create New Spool Request

The PRINTNEWSPOOL statement specifies whether to create a new spool request or append the spool request to an existing spool request with similar attributes (if any).

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTNEWSPOOL Y|N
```

Y

Creates new spool request.

N

Appends spool request to an existing spool request.

Notes:

- Use the PRINTNEWSPOOL statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTNEWSPOOL statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Append Spool Request to an Existing Request

This example appends the spool request to an existing spool request for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTNEWSPOOL N
```

PRINTPRIORITY Statement—Specify the Priority of the SAP Spool Request

The PRINTPRIORITY statement specifies the priority of the SAP spool request.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTPRIORITY priority
```

priority

Specifies the priority of the spool request. The priority is a number between 1 (highest priority) and 9 (lowest priority).

Limits: 1-9

Notes:

- Use the PRINTPRIORITY statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTPRIORITY statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify Spool Request Priority

This example sets the spool request to the highest priority for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTPRIORITY 1
```

PRINTPW Statement—Specify SAP Authorization String for Viewing Print Spool List

The PRINTPW statement specifies the SAP authorization string for viewing the print spool list.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTPW auth_string
```

auth_string

Specifies the required authorization string. The value corresponds to the SAPGUI Authorization field on the Background Print Parameters dialog.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTPW statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTPW statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Authorization String

This example specifies authstring as the authorization string for viewing the print spool list for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   PRINTPW authstring
```

PRINTREL Statement—Specify Deletion of Spool Request

The PRINTREL statement specifies whether to delete the spool request associated with the ABAP after printing.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTREL YES|NO
```

YES

Deletes the spool request after printing.

NO

Does not delete the spool request after printing.

Notes:

- Use the PRINTREL statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTREL statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- PRINTREL corresponds to the SAPGUI Spool options, Delete after output field on the Background Print Parameters dialog.

Example: Delete the Spool Request After Printing

This example deletes the spool request for the BTCTEST ABAP after printing:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   PRINTREL YES
```

PRINTREQTYPE Statement—Specify Print Request Type

The PRINTREQTYPE statement specifies the print request type.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTREQTYPE request_type
```

request_type

Specifies the print request type.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTREQTYPE statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTREQTYPE statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Print Request Type

This example specifies prtreq12 as the print request type for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTREQTYPE prtreq12
```

PRINTSPOOLNAME Statement—Specify Name of SAP Print Spool

The PRINTSPOOLNAME statement specifies the name of the SAP print spool.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PRINTSPOOLNAME spool_name
```

spool_name

Specifies the name of the print spool.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the PRINTSPOOLNAME statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the PRINTSPOOLNAME statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Print Spool Name

This example specifies prtspool12 as the print spool name for the RSPARAM ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME RSPARAM
   PRINTDEST LP01
   PRINTSPOOLNAME prtspool12
```

PRINTSTYLE Statement—Specify an Oracle Applications Print Style

The PRINTSTYLE statement specifies an Oracle Applications print style.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
PRINTSTYLE 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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default print style for all Oracle E-Business Suite Request Set or Single Request jobs by setting the oa.default.printStyle parameter in the agent's agentparm.txt file.
- The PRINTSTYLE statement overrides the default setting specified in the oa.default.printStyle parameter 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 PROGPRINTSTYLE statement. This statement overrides the PRINTSTYLE statement for that particular program.

Example: Specify an Oracle Applications Print Style

In this example, the job specifies three copies to be printed on the Q_DEVELOPMENT printer in PORTRAIT style.

```
AGENT CYBOA
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PROGRAM XXCOFI
PRINTER Q_DEVELOPMENT
PRINTSTYLE PORTRAIT
PRINTCOPIES 3
```

PRIVPROTOCOL Statement—Specify the SNMP v3 Privacy Protocol

The PRIVPROTOCOL statement specifies the SNMP v3 privacy protocol to use. This statement is ignored if your SNMP version is v1 or v2.

Supported Job Types

If your SNMP version is v3, this statement is required for the following job types:

- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
PRIVPROTOCOL DES|AES
```

DES

Specifies the Data Encryption Standard (DES) privacy protocol. DES uses a 16-character encryption key.

AES

Specifies the Advanced Encryption Standard (AES) privacy protocol. AES uses a 32-character encryption key. AES is the algorithm required by U.S. Government organizations to protect sensitive (unclassified) information (FIPS-140-2 compliance).

Example: Specify Privacy Protocol in an SNMP Value Set Job

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

PROCESS Statement—Specify the Process to be Monitored (Process Monitoring Jobs)

The PROCESS statement specifies the name of the process to be monitored.

Supported Job Type

This statement is required for the Process Monitoring job type (see page 126).

Syntax

This statement has the following format:

```
PROCESS process_name
```

process_name

Specifies the name of the process to be monitored.

Limits: Up to 256 characters; case-sensitive

UNIX: Specify a PID or process name.

Windows: Specify a full path or process name.

i5/OS:

Specify the following format:

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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you have multiple processes on a system with the same process name, the job completes successfully as follows:
  - STATUS RUNNING NOW—The job completes successfully if at least one of the processes with the specified name is running.
  - STATUS RUNNING WAIT—The job completes successfully when at least one of the processes with the specified name starts running.
  - STATUS STOPPED NOW—The job completes successfully if all of the processes with the specified name are stopped.
  - STATUS STOPPED WAIT—The job completes successfully when all of the processes with the specified name stops running.

i5/OS Notes:

- If you specify only the PID or process name, the agent searches the active UNIX workload for a matching PID or process name.
- 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 only a single asterisk and no other characters. To specify all possible names, use the special value *ALL.

Example: Monitor the Agent Process on UNIX Using a Process Name

This example monitors the cybAgent process on a UNIX computer. The job checks the process status immediately and completes successfully if the process is running.

```
AGENT UNIXAGENT
PROCESS cybAgent
STATUS RUNNING NOW
```

Example: Monitor the Agent Process on UNIX Using a PID

Suppose that the UNIXAGENT computer runs multiple agents. To monitor a specific agent process, specify the PID of that agent process. The job checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails.

```
AGENT UNIXAGENT
PROCESS 2568
STATUS RUNNING NOW
```

Example: Monitor the Agent Process on Windows

This example monitors the cybAgent.exe process on a Windows computer. The job checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails.

```
AGENT WINAGENT
PROCESS cybAgent.exe
STATUS RUNNING NOW
```

Example: Monitor the Agent Process on Windows Using a Full Path

Suppose that the WINAGENT computer runs multiple agents. To monitor the cybAgent.exe process, the full path to that process is specified. The job checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails. The text string is enclosed in single quotes because it contains a space.

```
AGENT WINAGENT
PROCESS 'c:\Program files\agentdir\cybAgent.exe'
STATUS RUNNING NOW
```

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. If the process is not running, the job fails.

```
PROCESS 123456/CYBESPU/CYBAGENT
STATUS RUNNING 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.

```
AGENT I5AG
PROCESS *ALL/JDOE/CALC*
STATUS STOPPED WAIT
```

PROCESS Statement—Specify the Process Status to Monitor (SAP Jobs)

PROCESS is an alias of PROCESSMONITOR (see page 486).

PROCESS_NAME Statement—Specify a PeopleSoft Process to Run

PROCESS_NAME is an alias of PROCESSNAME (see page 488).

PROCESS_PRIORITY Statement—Specify the Process Priority of a UNIX or i5/OS Job

The PROCESS_PRIORITY statement specifies the process priority of a UNIX or 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.

Supported Job Types

This statement is optional for the following job types:

- i5/OS (see page 113)
- UNIX (see page 190)

Syntax

This statement has the following format:

```
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.

BELOW_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.

Notes:

- If you do not specify the process priority in the job definition, the job's process priority is set to NORMAL by default.
- You can only set a UNIX job's process priority to a level above normal if the job runs on a computer with the agent started by the root account. If the agent is not started by root and you set the process priority to a level above normal, the job runs with the normal process priority.
- 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: Set the Process Priority for a UNIX Job

This example sets the process priority for a UNIX job to ABOVE_NORMAL:

```
AGENT LINUXAG
SCRIPTNAME /payroll/sort
PROCESS_PRIORITY ABOVE_NORMAL
```

Example: Set the Process Priority for an i5/OS Job

This example sets the process priority for an i5/OS job to HIGH:

```
AGENT I5AG
CLPNAME PAYROLL
PROCESS_PRIORITY HIGH
```

PROCESS_TYPE Statement—Specify the Type of PeopleSoft Process to Run

PROCESS_TYPE is an alias of PROCESSTYPE (see page 489).

PROCESSCHAIN Statement—Identify Name of Business Warehouse Process Chain on SAP System

PROCESSCHAIN is an alias of CHAIN (see page 277).

PROCESSMONITOR Statement—Specify the SAP Process Status to Monitor

The PROCESSMONITOR statement specifies the SAP process status to monitor.

Supported Job Type

This statement is required for the SAP Process Monitor job type (see page 165).

Syntax

This statement has the following format:

```
PROCESSMONITOR STATUS(WAITING|RUNNING|STOPPED)
    [USER(user)]
    [CLIENT(client)]
    [CONTINUOUS(alertid)]
    [TYPE(DIA|UPD|ENQ|BGD|SPO|UP2)]
```

STATUS

Specifies the SAP process status to monitor. Options are the following:

- RUNNING
- STOPPED
- WAITING

Note: If you specify WAITING, the CLIENT and USER operands do not apply.

USER(*user*)

(Optional) 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 32 valid SAP characters; case-sensitive; it cannot contain delimiters (such as spaces)

CLIENT(*client*)

(Optional) Specifies the client monitored for the specified process.

Limits: Up to 3 numeric digits

Example: 800

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified process type and status are detected. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified process type and status are detected.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

TYPE

(Optional) Specifies the SAP business process type to monitor. Options are the following:

- BGD—background
- DIA—dialog
- ENQ—lock
- SPO—spool
- UPD—update
- UP2—update2

Note: PROCESS is an alias of PROCESSMONITOR.

Example: Monitor for Jobs Running in the Background

This example monitors for running background jobs and checks for the SAP user named USER14:

```
SAPJOBNAME PMTEST
AGENT SAPHTAGENT
PROCESSMONITOR USER(USER14) CLIENT(800) CONTINUOUS(alt1) +
  STATUS(RUNNING) TYPE(BGD)
```

PROCESSNAME Statement—Specify a PeopleSoft Process to Run

The PROCESSNAME statement specifies the name of the PeopleSoft process that the job runs.

Supported Job Type

This statement is required for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
PROCESSNAME 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_PRCSDEFN table.

Limits: Up to 12 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROCESS_NAME is an alias of PROCESSNAME.

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.

```
AGENT PS_NY
PROCESSNAME AEMINTEST
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

PROCESSTYPE Statement—Specify the Type of PeopleSoft Process to Run

The PROCESSTYPE statement specifies the type of PeopleSoft prcoess that the job runs.

Supported Job Type

This statement is required for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
PROCESSTYPE 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 32 characters; case-sensitive

Options include the following:

- Application Engine
- COBOL
- Crystal reports
- n/Vision
- SQR

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROCESS_TYPE is an alias of PROCESSTYPE.

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.

```
AGENT PS_TOR
PROCESSNAME AEMINTEST
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

PROGARGS Statement—Define Argument Values to Pass to a Request Set Program

The PROGARGS statement defines the argument values to pass to an Oracle Applications request set program.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGARGS argument[,argument...]
```

argument

Defines each argument value to pass to an Oracle Applications request set program. In Oracle Applications, arguments are part of the program definition and are found in the Concurrent Program Parameters dialog.

Notes:

- Do not add a space after a comma, unless the argument value starts with a space.
- To use a default value defined in Oracle Applications, enter one comma in the argument position and specify Y in the ARGDEFAULTS statement. Do not add spaces between commas.
- The entire value can be up to 4070 characters. If your list of arguments is over the total maximum of 4070 characters, split it into multiple PROGARGS statements. These multiple statements will be concatenated using commas.
- You can specify up to 100 PROGARGS statements per request set program.
- PROGARGS applies to a single program in a request set.
- Place PROGARGS after a PROGDATA statement.

Example: Pass Argument Values to Request Set Programs

In this example, ARGDEFAULTS is set to Y. The first PROGARGS statement passes a value for the second argument, and the agent passes defaults for all others. The second PROGARGS statement passes values for the first, third, and fifth arguments, and the agent passes defaults for all others.

```
AGENT CYBOA
REQUESTSET 'EXTRACTS'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
PROGDATA 1
   PROGARGS ,X45S
PROGDATA 2
   PROGARGS X22F,,X56R143,,X23T
```

Example: Pass Date Arguments to a Request Set Program

In this example, five arguments are passed to the first program in the request set. For readability, the arguments are put into three PROGARGS statements, which will be concatenated using commas.

```
APPLDISPLNAME 'Customer Intelligence'
REQSET FNDRSSUB1587
OAUSER SYSADMIN
RESPNAME 'System Administrator'
PROGDATA
   PROGARGS 2004/10/04 00:00:00
   PROGARGS 2004/10/04 00:00:00
   PROGARGS N,LIFE_CYCLE,1468
ARGDEFAULTS N
```

PROGCOPIES Statement—Specify the Number of Copies to Print for a Request Set Program

PROGCOPIES is an alias of PROGPRINTCOPIES (see page 498).

PROGDATA Statement—Identify an Oracle Applications Program

The PROGDATA statement identifies a program in an Oracle Applications request set by order number. Use the PROGDATA statement to identify programs for the purpose of changing pre-set statement values.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGDATA [program_number]
```

program_number

(Optional) Specifies the order number of a program in an Oracle Applications request set. The *program_number* value is a fullword integer and as such the

maximum is 2**32 - 1 (2**32-1 means 2 power 32 minus 1, that is, 2 multiplied 32 times minus 1). The actual maximum number is 4294967295.

Limits: Integer

Note: If no number is specified, the first program is treated as number one, and subsequent programs are treated as having the previous program number plus one.

Notes:

- To specify data for multiple programs in a request set, define a separate PROGDATA statement for each program.
- You can use the following program-specific statements after a PROGDATA statement:
  - PROGARGS
  - PROGNOTIFYDUSERS
  - PROGNOTIFYUSERS
  - PROGOUTPUTFORMAT
  - PROGPRINTCOPIES
  - PROGPRINTER
  - PROGPRINTSTYLE
  - PROGQUOTEDEFAULT
  - PROGSAVEOUTPUT
  - PROGTEMPLATELANG
  - PROGTEMPLATETERR
- The program-specific statements cannot be used unless preceded by a PROGDATA statement.
- PROGRAMDATA is an alias of PROGDATA.

Example: Override Default Number of Print Copies in Certain Programs

In the following job definition, the PRINTCOPIES statement sets the default number of copies to be printed for all programs to one copy. The first PROGPRINTCOPIES statement overrides this default number of copies for program 1 to five copies. Similarly, the number of copies for program 2 is set to seven copies and the number of copies for program 4 is set to two copies.

The third program is not defined in this example, so the default number of copies (one) is used.

```
APPL ACC018
REQUESTSET 'XX EXTRACTS'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PRINTER Q8
PRINTCOPIES 1
PRINTSTYLE LANDSCAPE
PROGDATA
   PROGPRINTCOPIES 5
PROGDATA
   PROGPRINTCOPIES 7
PROGDATA 4
   PROGPRINTCOPIES 2
```

PROGNOTIFYDUSERS Statement—Specify Users to Notify Using Displays Names

The PROGNOTIFYDUSERS statement specifies a list of users to notify when a program in a request set completes using display names.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGNOTIFYDUSERS (display_user[,display_user...])
```

display_user

Specifies each user by display name.

Limits: case-sensitive

Notes:

- The entire value can be up to 1024 characters.
- Specify multiple user names by separating them with commas or blank spaces.
- To specify a user name with delimiters (such as spaces), enclose the user name in single quotes.
- PROGNOTIFYDUSERS applies to a single program in a request set.
- Place PROGNOTIFYDUSERS after the PROGDATA statement.

Example: Run Multiple Programs with Different Arguments and Notification Users

Suppose that you want to define a Request Set job to run multiple programs on the CYBOA agent. The request set FNDCPRT_SET belongs to the Application Object Library application in Oracle E-Business Suite and runs different arguments and notification users for the first, second, and seventh programs. Resolved expressions in default values are quoted. The job runs under the SYSADMIN user with System Administrator responsibility.

```
AGENT CYBOA
REQUESTSET FNDCPRT_SET
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
QUOTEDEFAULT Y
PROGDATA
   PROGARGS ,,,X,
   PROGNOTIFYDUSERS ('System Administrator','Admission Administrator')
PROGDATA 2
   PROGARGS ,2,,4
   PROGNOTIFYUSERS ('ASGUEST')
PROGDATA 7
   PROGARGS ,,arg3
```

PROGNOTIFYUSERS Statement—Specify Users to Notify Using Short Names

The PROGNOTIFYDUSERS statement specifies a list of users to notify when a program in a request set completes using short names.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGNOTIFYUSERS (short_user[,short_user...])
```

short_user

Specifies each user by short name.

Limits: case-sensitive

Notes:

- The entire value can be up to 1024 characters.
- Specify multiple user names by separating them with commas or blank spaces.
- To specify a user name with delimiters (such as spaces), enclose the user name in single quotes.
- PROGNOTIFYUSERS applies to a single program in a request set.
- Place PROGNOTIFYUSERS after the PROGDATA statement.

Example: Run Multiple Programs with Different Arguments and Notification Users

Suppose that you want to define a Request Set job to run multiple programs on the CYBOA agent. The request set FNDCPRT_SET belongs to the Application Object Library application in Oracle E-Business Suite and runs different arguments and notification users for the first, second, and seventh programs. Resolved expressions in default values are quoted. The job runs under the SYSADMIN user with System Administrator responsibility.

```
AGENT CYBOA
REQUESTSET FNDCPRT_SET
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
QUOTEDEFAULT Y
PROGDATA
   PROGARGS ,,,X,
   PROGNOTIFYDUSERS ('System Administrator','Admission Administrator')
PROGDATA 2
   PROGARGS ,2,,4
   PROGNOTIFYUSERS ('ASGUEST')
PROGDATA 7
   PROGARGS ,,arg3
```

PROGOUTPUTFORMAT Statement—Specify the Output Format for a Program in a Request Set

The PROGOUTPUTFORMAT statement specifies the output format for a program in a request set.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGOUTPUTFORMAT output_format
```

output_format

Specifies the output format for a program in a request set.

Limits: Up to ten characters; case-sensitive

Notes:

- PROGOUTPUTFORMAT applies to a single program in a request set. It overrides the OUTPUTFORMAT statement.
- Place PROGOUTPUTFORMAT after the PROGDATA statement.

Example: Specify Different Output Formats for Different Programs

In this example, both programs have template territory as United Kingdom (UK) and template language as German (DE). The first program has output format as XML, and the second program has output format as Excel.

```
AGENT OAAGENT
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQSET FNDRSSUB932
APPLDISPLNAME 'Application Object Library'
ARGDEFAULTS Yes
OUTPUTFORMAT XML
PROGDATA 1
    PROGARGS a
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
PROGDATA 2
    PROGARGS b
    PROGOUTPUTFORMAT Excel
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
```

PROGPRINTCOPIES Statement—Specify the Number of Copies to Print for a Request Set Program

The PROGPRINTCOPIES statement specifies the number of copies to print for an Oracle Applications request set program.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGPRINTCOPIES 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 six numeric digits

Example: 100

Notes:

- PROGPRINTCOPIES applies to a single program in a request set. It overrides the PRINTCOPIES statement.
- Place PROGPRINTCOPIES after a PROGDATA statement.
- PROGCOPIES is an alias of PROGPRINTCOPIES.

Example: Print a Different Number of Copies for Different Programs

In this example, five copies are printed for program 1 and eight for program 2.

```
AGENT CYBOA
REQUESTSET 'XX EXTRACTS'
APPLDISPLNAME 'Oracle Bill of Materials'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS N
PROGDATA 1
   PROGARGS ,12345,2,,3223,2,,1
   PROGPRINTER Q_DEVELOPMENT
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 5
   PROGSAVEOUTPUT YES
PROGDATA 2
   PROGARGS 12345,,,,1,1755,,-1,,
   PROGPRINTER Q_DEVELOPMENT
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 8
   PROGSAVEOUTPUT YES
```

PROGPRINTER Statement—Specify a Printer for a Request Set Program

The PROGPRINTER statement specifies the name of a printer to be used by an Oracle Applications request set program.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGPRINTER 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

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROGPRINTER applies to a single program in a request set. It overrides the PRINTER statement.
- Place PROGPRINTER after the PROGDATA statement.

Example: Specify a Printer for a Program in a Request Set

In this example, the PRINTER statement sets the default printer for this request set to Q_8. The first program has a PROGPRINTER statement, so its output is sent to the Q_DEVELOPMENT printer. The second program does not have a PROGPRINTER statement, so its output is sent to the Q_8 printer.

```
AGENT CYBOA
REQUESTSET 'EXTRACTS'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PRINTER Q_8
PROGDATA 1
   PROGARGS ,12345,2,,3223,2,,1
   PROGPRINTER Q_DEVELOPMENT
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 5
   PROGSAVEOUTPUT YES
PROGDATA 2
   PROGARGS 12345,,,,1,1755,,-1,,
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 8
   PROGSAVEOUTPUT YES
```

PROGPRINTSTYLE Statement—Specify Print Style for Request Set Program

The PROGPRINTSTYLE statement specifies a print style to be used by an Oracle Applications request set program.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGPRINTSTYLE 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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROGPRINTSTYLE applies to a single program in a request set. It overrides the PRINTSTYLE statement.
- Place PROGPRINTSTYLE after a PROGDATA statement.

Example: Specify a Print Style for a Program in a Request Set

In this example, the PRINTSTYLE statement sets the default printer for this request set to LANDSCAPE. The first program has a PROGPRINTSTYLE statement, so it uses the PORTRAIT print style. The second program does not have a PROGPRINTSTYLE statement, so it uses the default LANDSCAPE style.

```
AGENT CYBOA
REQUESTSET 'XX EXTRACTS'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PRINTERSTYLE LANDSCAPE
PROGDATA 1
   PROGARGS ,12345,2,,3223,2,,1
   PROGPRINTER Q_DEVELOPMENT
   PROGPRINTSTYLE PORTRAIT
   PROGPRINTCOPIES 5
   PROGSAVEOUTPUT YES
PROGDATA 2
   PROGARGS 12345,,,,1,1755,,-1,,
   PROGPRINTER Q_DEVELOPMENT
   PROGPRINTCOPIES 8
   PROGSAVEOUTPUT YES
```

PROGQUOTEDEFAULT Statement—Specify Whether to Quote Resolved Expressions in Default Values

The PROGQUOTEDEFAULT statement specifies whether to quote resolved expressions in default values for a program in a request set.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGQUOTEDEFAULT Y|N
```

Y

Specifies that resolved expressions in default values are quoted.

N

Specifies that resolved expressions in default values are not quoted. This is the default.

Notes:

- PROGQUOTEDEFAULT applies to a single program in a request set. It overrides the QUOTEDEFAULT statement.
- Place PROGQUOTEDEFAULT after the PROGDATA statement.

Example: Quote Resolved Expressions in Default Values

In this example, resolved expressions in default values are quoted in the second program.

```
AGENT OAAGENT
REQUESTSET FNDCPRT_SET
OAUSER SYSADMIN
RESPNAME 'System Administrator'
DISPLNAME 'Application Object Library'
PROGDATA 1
    PROGARGS ,,,X,
    PROGQUOTEDEFAULT N
PROGDATA 2
    PROGARGS ,2,,4
    PROGQUOTEDEFAULT Y
PROGDATA 7
    PROGARGS ,,,arg3
```

PROGRAM Statement—Specify the Short Name of an Oracle Applications Single Request Program

The PROGRAM statement specifies the short name of an Oracle Applications single request program.

Supported Job Type

If the PROGRAMDISPL statement is not used, this statement is required for the Oracle E-Business Suite Single Request job type (see page 142).

Note: All Oracle E-Business Suite Single Request job definitions require a PROGRAM statement or a PROGRAMDISPL statement. Use the PROGRAMDISPL statement to specify the display name of the program and the PROGRAM statement to specify the short name of the program.

Syntax

This statement has the following format:

```
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

Note: The value of the oracle_program_name_type attribute must be SHORT.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the Name of a Single Request Program

In this example, the single request program is identified by its short name.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

PROGRAMDATA Statement—Identify an Oracle Applications Program

PROGRAMDATA is an alias of PROGDATA (see page 492).

PROGRAMDISPL Statement—Specify the Display Name of an Oracle Applications Single Request Program

The PROGRAM statement specifies the display name of an Oracle Applications single request program.

Supported Job Type

If the PROGRAM statement is not used, this statement is required for the Oracle E-Business Suite Single Request job type (see page 142).

Note: All Oracle E-Business Suite Single Request job definitions require a PROGRAM statement or a PROGRAMDISPL statement. Use the PROGRAMDISPL statement to specify the display name of the program and the PROGRAM statement to specify the short name of the program.

Syntax

This statement has the following format:

```
PROGRAMDISPL program_display_name
```

program_display_name

Specifies the display name of the Oracle Applications single request program.

Limits: Up to 240 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the Name of a Single Request Program

In this example, the single request program is identified by its display name.

```
AGENT CYBOA
PROGRAMDISPL 'Load AP Invoice Source Dimension'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

PROGSAVEOUTPUT Statement—Specify Whether to Save Output from a Request Set Program

The PROGSAVEOUTPUT statement specifies whether to save the output from an Oracle Applications request set program.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGSAVEOUTPUT 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.

Notes:

- PROGSAVEOUTPUT applies to a single program in a request set. PROGSAVEOUTPUT overrides the SAVEOUTPUT statement.
- Place PROGSAVEOUTPUT after the PROGDATA statement.
- The PROGSAVEOUTPUT statement is ignored when Y or N is omitted.

Example: Save Output for Only One Oracle Applications Program in a Request Set

In the following example, the PROGSAVEOUTPUT statement is defined as Y for the first program and N for the second. When the job executes, an output file is created by Oracle Applications and is saved for program 1, but not for program 2.

```
AGENT CYBOA
REQUESTSET 'XX EXTRACTS'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
PROGDATA 1
   PROGARGS ,12345,2,,3223,2,,1
   PROGSAVEOUTPUT YES
PROGDATA 2
   PROGARGS 12345,,,,1,1755,,-1,,
   PROGSAVEOUTPUT N
```

PROGTEMPLATELANG Statement—Specify the Template Language for a Program in a Request Set

The PROGTEMPLATELANG statement specifies the template language for a program in a request set.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGTEMPLATELANG template_language
```

template_language

Specifies the template language for a program in a request set.

Limits: Up to ten characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROGTEMPLATELANG applies to a single program in a request set. It overrides the TEMPLATELANG statement.
- Place PROGTEMPLATELANG after a PROGDATA statement.

Example: Specify Template Language for Different Programs

In this example, both programs have template territory as United Kingdom (UK) and template language as German (DE). The first program has output format as XML, and the second program has output format as Excel.

```
AGENT OAAGENT
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQSET FNDRSSUB932
APPLDISPLNAME 'Application Object Library'
ARGDEFAULTS Yes
OUTPUTFORMAT XML
PROGDATA 1
    PROGARGS a
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
PROGDATA 2
    PROGARGS b
    PROGOUTPUTFORMAT Excel
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
```

PROGTEMPLATETERR Statement—Specify the Template Territory for a Program in a Request Set

The PROGTEMPLATETERR statement specifies the template territory for a program in a request set.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
PROGTEMPLATETERR template_territory
```

template_territory

Specifies the template territory for a program in a request set.

Limits: Up to ten characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- PROGTEMPLATETERR applies to a single program in a request set. It overrides the TEMPLATETERR statement.
- Place PROGTEMPLATETERR after a PROGDATA statement.

Example: Specify Template Territory for Different Programs

In this example, both programs have template territory as United Kingdom (UK) and template language as German (DE). The first program has output format as XML, and the second program has output format as Excel.

```
AGENT OAAGENT
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQSET FNDRSSUB932
APPLDISPLNAME 'Application Object Library'
ARGDEFAULTS Yes
OUTPUTFORMAT XML
PROGDATA 1
    PROGARGS a
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
PROGDATA 2
    PROGARGS b
    PROGOUTPUTFORMAT Excel
    PROGTEMPLATETERR UK
    PROGTEMPLATELANG DE
```

PROVIDER_URL Statement—Specify a Service Provider URL

PROVIDER_URL is an alias of LOCATION (see page 413).

PROXYDOMAIN Statement—Specify a Domain for Proxy Authentication

The PROXYDOMAIN statement specifies the domain for proxy authentication in an HTTP job. This statement is required for NTLM authentication.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
PROXYDOMAIN proxy_domain
```

proxy_domain

Specifies the domain for proxy authentication.

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 PROXYDOMAIN statement 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

PROXYHOST Statement—Specify a Proxy Host

The PROXYHOST statement specifies the proxy host name to use for the request in an HTTP job.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

PROXYHOST *proxy_host*

proxy_host

Specifies the proxy host name to use for the request.

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 PROXYHOST statement 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

PROXYORIGIN Statement—Specify an Origin Host Name for Proxy Authentication

The PROXYORIGIN statement specifies the origin host name for proxy authentication in an HTTP job. This statement is used for NTLM authentication.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
PROXYORIGIN proxy_origin
```

proxy_origin

Specifies the origin host name for proxy authentication.

Limits: Up to 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default origin host name for all HTTP jobs by setting the http.proxyOrigin parameter in the agent's agentparm.txt file.
- The PROXYORIGIN statement overrides the default origin 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

PROXYPORT Statement—Specify a Proxy Port

The PROXYPORT statement specifies the proxy port to use for the request in an HTTP job.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
PROXYPORT proxy_port
```

proxy_port

Specifies the proxy port to use for the request.

Default: 80 (Web Service RPC/Encoded) and http.proxyPort agent parameter, if specified (Web Service Document/Literal)

Limits: 0-65535

Notes:

- Your agent administrator can specify a default proxy port for all HTTP jobs by setting the http.proxyPort parameter in the agent's agentparm.txt file.
- The PROXYPORT statement overrides the default proxy port 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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

PROXYUSER Statement—Specify a User for Proxy Authentication

The PROXYUSER statement specifies the user name required for proxy authentication and, optionally, the user's domain in an HTTP job. The scheduling manager uses the user name, domain, agent name, and the job type to select the user's password in the password repository.

Supported Job Type

This statement is optional for the HTTP job type (see page 42).

Syntax

This statement has the following format:

PROXYUSER *proxy_user* [DOMAIN(*domain*)]

proxy_user

Specifies the user name required for proxy authentication.

Limits: Up to 128 characters; case-sensitive; it cannot contain delimiters (such as spaces)

DOMAIN(*domain*)

(Optional) Specifies the name of the user's domain.

Limits: Up to 64 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 PROXYUSER statement overrides the default proxy user name specified in the agent's agentparm.txt.
- Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

PSOPRID Statement—Specify Operator ID for a PeopleSoft Report

The PSOPRID statement specifies the operator ID under whose authority the PeopleSoft report runs.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
PSOPRID ID
```

ID

Specifies the operator ID under whose authority the PeopleSoft report runs. The scheduling manager supplies the corresponding password from its password facility.

Limits: Up to 32 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- The operator ID is validated against the corresponding value in the PeopleSoft database.
- If you do not specify this statement in the job definition, a default operator ID must be defined in the ps.default.oprId parameter in the agent's agentparm.txt file. Otherwise, the job fails.
- The PSOPRID statement overrides the default operator ID specified in the agent's agentparm.txt file.
- Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

Example: Specify an Operator ID the Report Runs Under

In this example, the DDAUDIT report runs under the authority of the VP1 operator ID.

```
AGENT PSAGENT
PROCESSNAME DDAUDIT
PSOPRID VP1
PROCESSTYPE 'SQR Report'
RUNCONTROLID TEST
OUTDESTTYPE FILE
OUTDESTFORMAT TXT
```

PUTINOUTBOX Statement—Specify Whether to Save Outgoing Documents to the SAPoffice Outbox

The PUTINOUTBOX statement specifies whether to save outgoing documents to the SAPoffice outbox of the SAP user associated with the job.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
PUTINOUTBOX Y|N
```

Y

Saves outgoing documents to the outbox of the SAP user.

N

Does not save outgoing documents to the outbox of the SAP user.

Example: Save Outgoing Mail to the SAPoffice Outbox

This example sends outgoing mail to the SAPoffice outbox of the SAP user J01Prod:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
RECIPIENTTYPE SU
SPOOLRECIPIENT CA
PUTINOUTBOX Y
ABAPNAME RSPARAM
```

QUOTEDEFAULT Statement—Specify Whether to Quote Resolved Expressions in Default Values

The QUOTEDEFAULT statement specifies whether to quote resolved expressions in default values.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
QUOTEDEFAULT Y|N
```

Y

Specifies that resolved expressions in default values are quoted.

N

Specifies that resolved expressions in default values are not quoted. This is the default.

Note: In an Oracle E-Business Suite Request Set job, you can specify whether to quote resolved expressions in default values in the request set by using the PROGQUOTEDEFAULT statement. This statement overrides the QUOTEDEFAULT statement for that particular program.

Example: Quote Resolved Expressions in Default Values

Suppose that you want to define a Request Set job to run multiple programs on the CYBOA agent. The request set FNDCPRT_SET belongs to the Application Object Library application in Oracle E-Business Suite and runs different arguments and notification users for the first, second, and seventh programs. Resolved expressions in default values are quoted. The job runs under the SYSADMIN user with System Administrator responsibility.

```
AGENT CYBOA
REQUESTSET FNDCPRT_SET
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
ARGDEFAULTS Y
QUOTEDEFAULT Y
PROGDATA
   PROGARGS ,,,X,
   PROGNOTIFYDUSERS ('System Administrator','Admission Administrator')
PROGDATA 2
   PROGARGS ,2,,4
   PROGNOTIFYUSERS ('ASGUEST')
PROGDATA 7
   PROGARGS ,,arg3
```

RANGE Statement—Specify Search Range

RANGE is an alias of SEARCHRANGE (see page 566).

RECIPIENT Statement—Specify Recipient Name for SAP Cover Page

The RECIPIENT statement specifies the spool request recipient's name to appear on the SAP cover page.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENT user_name
```

user_name

Specifies the spool request recipient's name to appear on the SAP cover page. The value corresponds to the SAPGUI Cover sheets options, Recipient field on the Background Print Parameters dialog.

Limits: Up to 12 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the RECIPIENT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the RECIPIENT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Recipient's Name to Appear on the SAP Cover Page

In this example, Recipient1 appears as the recipient's name on the SAP cover page.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   RECIPIENT Recipient1
```

RECIPIENTBCC Statement—Specify Whether to Send a Blind Carbon Copy of the SAP Spool

The RECIPIENTBCC statement specifies whether to send a blind carbon copy (bcc) of the SAP spool to spool recipients.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTBCC Y|N
```

Y

Sends a blind carbon copy of the spool to spool recipients.

N

Does not send a blind carbon copy of the spool to spool recipients. This is the default.

Note: If you use a RECIPIENTCC statement and a RECIPIENTBCC statement in a job, only one statement can be set to Y.

Example: Send a Blind Carbon Copy of the Spool File

This example sends a blind carbon copy of the spool file to the users on the payroll distribution list:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE INT
RECIPIENTBCC Y
SPOOLRECIPIENT payroll@company1.com
ABAPNAME BTCTEST
```

RECIPIENTCC Statement—Specify Whether to Send a Carbon Copy of the SAP Spool

The RECIPIENTCC statement specifies whether to send a carbon copy (cc) of the SAP spool to spool recipients.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTCC Y|N
```

Y

Sends a carbon copy of the SAP spool to spool recipients.

N

Does not send a carbon copy of the SAP spool to spool recipients. This is the default.

Note: If you use a RECIPIENTCC statement and a RECIPIENTBCC statement in a job, only one statement can be set to Y.

Example: Send a Carbon Copy of the Spool File

This example sends a carbon copy of the spool file to the users on the payroll distribution list:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE INT
RECIPIENTCC Y
SPOOLRECIPIENT payroll@company1.com
ABAPNAME BTCTEST
```

RECIPIENTEXPRESS Statement—Specify Whether to Send the SAP Spool Using Express Delivery

The RECIPIENTEXPRESS statement specifies whether to send the SAP spool using express delivery.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTEXPRESS Y|N
```

Y

Sends the spool using express delivery. Recipients receive a message instantly if they are logged into SAP.

N

Does not send the spool using express delivery. This is the default.

Note: The RECIIENTEXPRESS statement applies only when the RECIPIENTTYPE statement is set to DLI or SU.

Example: Send the Spool File to an SAP User Using Express Delivery

This example sends the spool file to the SAP user J01Prod using express delivery. If J01Prod is logged into the SAP system, J01PROD will receive a message that the spool file has been sent.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
RECIPIENTEXPRESS Y
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

RECIPIENTFORWARD Statement—Allow Forwarding of SAP Spool

The RECIPIENTFORWARD statement specifies whether to allow spool recipients to forward the SAP spool.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTFORWARD Y|N
```

Y

Allows spool recipients to forward the spool.

N

Does not allow spool recipients to forward the spool.

Note: The RECIIENTFORWARD statement applies only when the RECIPIENTTYPE statement is set to DLI or SU.

Example: Allow an SAP User to Forward the Spool File

This example allows the SAP user J01Prod to forward the spool file:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
RECIPIENTFORWARD Y
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

RECIPIENTPRINT Statement—Specify Whether to Allow Printing of the SAP Spool

The RECIPIENTPRINT statement specifies whether to allow spool recipients to print the SAP spool.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTPRINT Y|N
```

Y

Allow spool recipients to print the spool.

N

Does not allow spool recipients to print the spool.

Note: The RECIPIENTPRINT statement applies only when the RECIPIENTPRINT statement is set to DLI or SU.

Example: Allow an SAP User to Print the Spool File

This example allows the SAP user J01Prod to print the spool file:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
RECIPIENTPRINT Y
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

RECIPIENTTYPE Statement—Specify the Recipient Type for the SAP Spool

The RECIPIENTTYPE statement specifies the recipient type for the SAP spool.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RECIPIENTTYPE SU|DLI|INT
```

SU

Specifies SAP user as the recipient type.

DLI

Specifies SAPoffice distribution list as the recipient type.

INT

Specifies email address as the recipient type. You can also use the operand MAIL.

Example: Specify SAP User as the Recipient Type

This example specifies that the SAP user J01Prod receives the spool file upon job completion:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

REMOTECLASSNAME Statement—Specify a Remote Class Name

The REMOTECLASSNAME statement specifies the naming class of the Java object you want to invoke a method on in an RMI job.

Supported Job Type

This statement is required for the RMI job type (see page 62).

Syntax

This statement has the following format:

```
REMOTECLASSNAME 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 1024 characters; case-sensitive

Example: rmi://smith:5000/test

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
REMOTECLASSNAME rmi://remotehost/Test
METHOD startserver
PARAMETER TYPE(String) VALUE(now)
```

REMOTEDIR Statement—Specify a Remote Directory

The REMOTEDIR statement specifies the directory on a remote server involved in a secure file transfer.

Supported Job Types

This statement is required for the following job types:

- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
REMOTEDIR 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). Since the LOCALNAME statement contains a wildcard, the REMOTENAME statement is not specified.

```
AGENT WINAGENT
SERVERADDR aixunix
TRANSFERDIRECTION UPLOAD
USER causer
REMOTEDIR /u1/build/uploaded
LOCALNAME c:\temp\upload\*
```

REMOTEFILENAME Statement—Specify Remote Filename (FTP Job)

The REMOTEFILENAME statement specifies the name of one or more files on a remove server involved in an FTP transfer.

Supported Job Types

This statement is required for the FTP job type (see page 96).

Syntax

This statement has the following format:

```
REMOTEFILENAME file_name[; file_name...]
```

file_name

Specifies the source location of the file (if downloading) or the destination of the file (if uploading).

Limits: The total of all files specified cannot exceed 256 characters; case-sensitive

UNIX/Windows:

- If you are uploading a file, specify the full path and file name.
- 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, use a forward slash at the beginning of the path statement and between the directories and file name. For example:

  ```
  remotefilename /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
  ```

Mainframe:

- By default, the data set high-level prefix is set to the user ID of the logged-on user.
- To specify a different high-level prefix, fully qualify the value and wrap double quotes around it, for example: "'SYS4.CA7.JCLLIB(CA07XX01)'".

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 number of files specified in the LOCALFILENAME and REMOTEFILENAME statements must match.
- REMOTENAME is an alias of REMOTEFILENAME.

Example: Download a Single File

This example downloads an ASCII file named textfile from the remote UNIX server to the /export/home/ftpfiles/ftpdata directory on the agent computer:

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /u1/test/ftpdata/textfile
LOCALFILENAME /export/home/ftpfiles/ftpdata/textfile_dn
```

Example: Download Multiple Files

This example downloads multiple ASCII files from a remote UNIX server. Note that the number of files listed in the REMOTEFILENAME and LOCALFILENAME statements must match. Single quotes enclose both lists to accommodate the line continuation character (+).

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME '/u1/test/scripts/echo;+
  /u1/test/scripts/echo1;+
  /u1/test/scripts/echo2'
LOCALFILENAME '/export/home/test/ftpdata/echo;+
  /export/home/test/ftpdata/echo_1;+
  /export/home/test/ftpdata/echo_2'
```

Example: Upload a Windows File to a Mainframe Data Set

This example uploads a file named TESTS to the mainframe. By default, the REMOTEFILENAME statement uses the user ID of the logged-on user for the data set high-level prefix.

Note: To specify a different high-level prefix, fully qualify the value of the REMOTEFILENAME statement and wrap double quotes around it.

```
AGENT WINAGENT
LOCALFILENAME 'C:\My Documents\Microsoft Office Directories+
WordPad\tests.txt'
REMOTEFILENAME ‘TESTS.DATA'
SERVERADDR MAINFRAME.COM
TRANSFERCODETYPE U
TRANSFERDIRECTION UPLOAD
USER causer
```

Example: Download a Mainframe Data Set to a Windows File

This example downloads the data set member CA07XX01 to a Windows computer. The value of the REMOTEFILENAME statement is fully qualified by wrapping double quotes around the single quotes.

```
AGENT WINAGENT
LOCALFILENAME 'C:\My Documents\File Transfers\ca07xx01.txt'
REMOTEFILENAME "'SYS4.CA7.JCLLIB(CA07XX01)'"
SERVERADDR MAINFRAME.COM
TRANSFERCODETYPE A
TRANSFERDIRECTION DOWNLOAD
USER lmla
```

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.

```
AGENT UNIXAGENT
USER test
SERVERADDR hpunsup
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /u1/test/scripts/r*
LOCALFILENAME /export/home/test/ftpdata/WC
```

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 REMOTEFILENAME and LOCALFILENAME statements. The remote Windows computer uses an agent as the FTP server. Its IP address is 172.16.0.0.

```
AGENT UNIXAGENT
USER test
SERVERADDR 172.16.0.0
SERVERPORT 123
TRANSFERDIRECTION U
TRANSFERCODETYPE A
REMOTEFILENAME /C:/TEMP/textfile
LOCALFILENAME /export/home/test/ftpdata/textfile
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5agent
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

REMOTENAME Statement—Specify Remote Filenames (FTP Jobs)

REMOTENAME is an alias of REMOTEFILENAME (see page 530).

REMOTENAME Statement—Specify a Remote File Name (Secure Copy and Secure FTP Jobs)

The REMOTENAME statement specifies a file on a remote server to be downloaded or uploaded in a secure file transfer.

Supported Job Types

This statement is required for the Secure Copy job type (see page 174).

This statement is optional for the Secure FTP job type (see page 175).

Note: For Secure FTP jobs, the REMOTENAME statement is required if the LOCALNAME statement does not contain a wildcard.

Syntax

This statement has the following format:

`REMOTENAME file_name`

file_name

Specifies the file's source location (if downloading) or the file's destination (if uploading).

Limits: Up to 256 characters; case-sensitive

Notes:

- For uploads, you must specify the file name without wildcards.
- For downloads, you can use wildcards for the file name if you are using the Secure FTP job. You cannot use wildcards if you are using the Secure Copy job. The asterisk (*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- You cannot rename files if wildcards are used.

Mainframe:

- By default, the data set high-level prefix is set to the user ID of the logged-on user.
- To specify a different high-level prefix, fully qualify the value and wrap double quotes around it, for example: "'SYS4.CA7.JCLLIB(CA07XX01)'".

Notes:

- Enclose values that contain delimiters (such as spaces) in single quotation marks.
- 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.

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):

```
AGENT WINAGENT
LOCALNAME C:\temp\install.log1
REMOTENAME install.log
REMOTEDIR /root
SERVERADDR chi-linux
SERVERPORT 22
TRANSFERDIRECTION DOWNLOAD
USER causer
```

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).

```
AGENT WINAGENT
TRANSFERDIRECTION UPLOAD
SERVERADDR hpsupport
REMOTEDIR /u/tmp
REMOTENAME logs.tar
LOCALNAME D:\temp\logs.tar
USER causer
```

REQSET Statement—Specify the Short Name of an Oracle Applications Request Set

REQSET is an alias of REQUESTSET (see page 538).

REQSETDISPL Statement—Specify the Display Name of an Oracle Applications Request Set

The REQUESTSETDISPL statement specifies the display name of an Oracle Applications request set.

Supported Job Type

If the REQUESTSET statement is not used, this statement is required for the Oracle E-Business Suite Request Set job type (see page 137).

Note: All Oracle E-Business Suite Request Set job definitions require a REQUESTSET statement or a REQSETDISPL statement. Use the REQSETDISPL statement to specify the display name of the request set and the REQUESTSET statement to specify the short name of the request set.

Syntax

This statement has the following format:

`REQSETDISPL` *`request_set_display_name`*

request_set_display_name

Specifies the display name of the Oracle Applications request set.

Limits: Up to 240 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the Name of a Request Set

In the following example, the request set is identified by its display name.

```
AGENT CYBOA
REQSETDISPL 'T4A Amendment Register Report CA'
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

REQUESTID Statement—Specify Request ID of the Oracle Applications Request to Copy

The REQUESTID statement specifies the request ID of the Oracle Applications request you want to copy. You can copy an existing single request defined on Oracle Applications and run it under the agent.

Note: In this release, you cannot copy an existing request set defined on Oracle Applications and run it under the agent.

Supported Job Type

This statement is required for the Oracle E-Business Suite Copy Job job type (see page 136).

Syntax

This statement has the following format:

```
REQUESTID request_id [REQUEST]
```

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: 493895459709656750

REQUEST

(Optional) Indicates that you want to copy an existing single request defined on Oracle Applications. This is the default.

Example: Copy a Single Request

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

REQUESTSET Statement—Specify the Short Name of an Oracle Applications Request Set

The REQUESTSET statement specifies the short name of an Oracle Applications request set.

Supported Job Type

If the REQSETDISPL statement is not used, this statement is required for the Oracle E-Business Suite Request Set job type (see page 137).

Note: All Oracle E-Business Suite Request Set job definitions require a REQUESTSET statement or a REQSETDISPL statement. Use the REQSETDISPL statement to specify the display name of the request set and the REQUESTSET statement to specify the short name of the request set.

Syntax

This statement has the following format:

```
REQUESTSET 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

Note: The value of the oracle_req_set_type attribute must be SHORT.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- REQSET is an alias of REQUESTSET.

Example: Specify the Name of a Request Set

In the following example, the request set is identified by its short name.

```
AGENT CYBOA
REQUESTSET DATA10
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
```

RESPNAME Statement—Specify an Oracle Applications Responsibility Name

The RESPNAME statement specifies an Oracle Applications responsibility name.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Copy Job (see page 136)
- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
RESPNAME 'responsibility_name'
```

responsibility_name

Specifies an Oracle Applications responsibility name.

Limits: Up to 100 characters; case-sensitive

Oracle E-Business Suite Single Request or Request Set Job Notes:

- If you do not specify this statement 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 RESPNAME statement overrides the default responsibility name specified in the agent's agentparm.txt file.

Oracle E-Business Suite Copy 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 OAUSER statement 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 RESPNAME statement 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 a Responsibility Name in an Oracle E-Business Suite Request Set Job

In the following example, the responsibility name for the BIS application is Business Intelligence Super User, Progress S&L.

```
AGENT CYBOA
REQUESTSET FNDRSSUB1310
OAUSER SYSADMIN
RESPNAME 'Business Intelligence Super User, Progress S&L'
APPLSHORTNAME BIS
```

Example: Override the Responsibility Name in an Oracle E-Business Suite Copy Job

Suppose that you want to copy an existing single request defined on Oracle Applications. In this example, the job copies the single request with request ID 2255470 and overrides the user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

```
AGENT CYB0A
OAUSER SYSADMIN
RESPNAME 'System Administrator'
REQUESTID 2255470
CHILDMONITOR Y
MONITORDELAY 60
```

RETURN Statement—Specify JDBC Data Type for a Return Value

RETURN is an alias of RETURN_DATA_TYPE (see page 540).

RETURN_DATA_TYPE Statement—Specify JDBC Data Type for a Return Value

The RETURN_DATA_TYPE statement specifies the JDBC data type for the value returned by a stored function.

Supported Job Type

This statement is optional for the Database Stored Procedure job type (see page 71).

Syntax

This statement has the following format:

```
RETURN_DATA_TYPE return_type
```

return_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) VARCHAR(x)	VARCHAR(x)	VARCHAR(x)
LONGVARCHAR		Not supported	Not supported	Not supported
NUMERIC	Number	NUMBER, 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
BIGINT	Number	Not supported	BIGINT	BIGINT
REAL	Number [dot Number]	REAL	REAL	REAL
FLOAT	Number [dot Number]	FLOAT	FLOAT	FLOAT
DOUBLE				DOUBLE
DATE	yyyy-mm-dd	DATE		DATE
DATE	String (for example, SYSDATE)			
TIME	hh:mm:ss			TIME

JDBC Data Type	Valid Input Format	Oracle	Microsoft SQL Server	DB2
TIMESTAMP	yyyy-mm-dd hh:mm:ss.fffffffff	TIMESTAMP		TIMESTAMP

Notes:

- You can run stored functions on Oracle or SQL Server using Stored Procedure jobs. The agent does not support stored functions (also named user-defined functions) on DB2.
- RETURN is an alias of RETURN_DATA_TYPE.

Example: Run Stored Function that Returns Type DECIMAL

In the following example, the stored function COUNT returns type DECIMAL.

```
AGENT CYBDB1
STORED_PROCEDURE COUNT
RETURN_DATA_TYPE DECIMAL
```

RETURNCLASS Statement—Specify a Return Class Name

The RETURNCLASS statement specifies the Java class name of the return value in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
RETURNCLASS return_class
```

return_class

Specifies the Java class name of the return value.

Limits: Up to 100 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

RETURNNAMESPACE Statement—Specify an XML Namespace

The RETURNNAMESPACE statement specifies the XML namespace for the RETURNXML statement in a Web Service job. If you omit this statement, the return namespace defaults to the target namespace specified by the TARGETNAMESPACE statement.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
RETURNNAMESPACE 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

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

RETURNXML Statement—Specify an XML Type

The RETURNXML statement specifies the XML type that maps to the RETURNCLASS statement in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
RETURNXML XML_type
```

XML_type

Specifies the XML type that maps to the Java class name of the return value.

Limits: Up to 100 characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

RFCDEST Statement—Specify SAP Destination

The RFCDEST statement specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Business Warehouse InfoPackage (see page 158)
- SAP Business Warehouse Process Chain (see page 159)
- SAP Data Archiving (see page 160)
- SAP Event Monitor (see page 162)
- SAP Job Copy (see page 164)
- SAP Process Monitor (see page 165)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
RFCDEST destination
```

destination

Specifies the destination for the RFC connection and gateway information for an SAP R/3 system. This is the destination specified in the connection properties file during installation.

Limits: Up to 20 valid SAP characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Specify the SAP Destination

This example runs the BTCTEST ABAP at the SAP destination, CTR3:

```
SAPJOBNAME SAPTEST
RFCDEST CTR3
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   RECIPIENT Recipient1
```

RUNCONTROLARGS Statement—Specify the Arguments for a PeopleSoft Run Control Table

The RUNCONTROLARGS statement specifies the run control arguments for a run control table on the PeopleSoft system. You can use this statement with the RUNCONTROLTABLE statement to add parameters to the run control table.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
RUNCONTROLARGS argument|'argument,argument...'
```

argument

Specifies a run control argument to add to the run control table.

Limits: Case-sensitive

Notes:

- If you specify a single argument containing spaces, enclose it in single quotation marks.
- You can specify multiple arguments. Separate each argument with a comma and enclose the entire list of arguments in single quotes.
- The entire value can be up to 4078 characters.
- If the number of values specified in the RUNCONTROLARGS statement 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 statement, for example:

  ```
  RUNCONTROLARGS 'BI_WF_0001,0,O,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.

Example: Add a Row of Process Parameters to a Run Control Table

In this example, the RUNCONTROLARGS statement specifies the run control arguments for the PS_AEREQUESTTBL run control table . The job inserts a new row in the PS_AEREQUESTTBL run control table with data from the run control arguments.

```
AGENT PSAGENT
PROCESSNAME XYZ
PROCESSTYPE 'Application Engine'
PSOPRID VP1
RUNCONTROLID test
RUNCONTROLTABLE PS_AEREQUESTTBL
RUNCONTROLARGS 'BI_WF_0001,0,O,N,0,N,,b\,sysdate'
```

The columns in the PS_AEREQUESTTBL run control table correspond to the run control arguments as follows:

Column	Value
AE_APPLID	BI_WF_0001
CURTEMPINSTANCE	0
PROCESS_FREQUENCY	O
AE_PROCESS_STATUS	N
PROCESS_INSTANCE	0
PROCESS_ORIG	N
LAST_RUN_DTM	null
MARKET	b\
ASOF_DT	2007-07-10 14:30:00

RUNCONTROLID Statement—Specify PeopleSoft Run Parameters

The RUNCONTROLID statement specifies a set of PeopleSoft run parameters for a PeopleSoft process.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
RUNCONTROLID 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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you do not specify this statement 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 RUNCONTROLID statement 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.

```
AGENT PS_TOR
PROCESSNAME AEMINTEST
PROCESSTYPE 'Application Engine'
RUNCONTROLID PS_ALL
```

Example: Check Children Status

This example runs the AEMINITEST process on the agent named PS_AGENT. The job completes successfully if the AEMINITEST process and all of its children complete successfully.

```
AGENT PS_AGENT
PROCESSNAME AEMINITEST
PROCESSTYPE 'Application Engine'
PSOPRID VP1
RUNCONTROLID HDL8010.*
```

RUNCONTROLTABLE Statement—Specify the Table that Contains the PeopleSoft Run Parameters

The RUNCONTROLTABLE statement specifies the name of the table that contains the run control parameters for a PeopleSoft process. You can use this statement with the RUNCONTROLARGS statement to add parameters to the run control table.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
RUNCONTROLTABLE control_table
```

control_table

Specifies the name of the table that contains the run parameters for a given PeopleSoft process.

Limits: Up to 100 characters; case-sensitive

Example: PS_AEREQUESTTBL

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Add a Row of Process Parameters to a Run Control Table

In this example, the RUNCONTROLTABLE statement 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.

```
AGENT PSAGENT
PROCESSNAME XYZ
PROCESSTYPE 'Application Engine'
PSOPRID VP1
RUNCONTROLID test
RUNCONTROLTABLE PS_AEREQUESTTBL
RUNCONTROLARGS 'BI_WF_0001,0,O,N,0,N,,b\,sysdate'
```

The columns in the PS_AEREQUESTTBL run control table correspond to the run control arguments as follows:

Column	Value
AE_APPLID	BI_WF_0001
CURTEMPINSTANCE	0
PROCESS_FREQUENCY	O
AE_PROCESS_STATUS	N
PROCESS_INSTANCE	0
PROCESS_ORIG	N
LAST_RUN_DTM	null
MARKET	b\
ASOF_DT	2007-07-10 14:30:00

SAPCLIENT Statement—Specify SAP Client

The SAPCLIENT statement specifies the SAP client within the SAP system.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Business Warehouse InfoPackage (see page 158)
- SAP Business Warehouse Process Chain (see page 159)
- SAP Data Archiving (see page 160)
- SAP Event Monitor (see page 162)
- SAP Job Copy (see page 164)
- SAP Process Monitor (see page 165)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPCLIENT 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

Example: 850

Notes:

- Your agent administrator can specify a default setting for all SAP jobs by setting the jco.client.client property in the connection properties file for the SAP system.
- The SAPCLIENT statement overrides the default setting specified in the jco.client.client property in the connection properties file for the SAP system.

Example: Specify the Client within the SAP System

This example runs the BTCTEST ABAP. The SAP client in the SAP system is 850.

```
SAPJOBNAME SAPTEST
SAPCLIENT 850
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPEMAILADDR Statement—Email SAP Spool File on Job Completion or Failure

The SAPEMAILADDR statement emails a copy of the SAP spool file to one or more email addresses upon job completion or failure.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPEMAILADDR address |'address,address,...'
```

address

Specifies the email address of a recipient on a distribution list. The value corresponds to the General data, Spool list recipient button on the Define Background Job dialog.

Limits: Up to 256 valid SAP characters; case-sensitive

Note: To specify multiple email addresses, separate each address with a comma and enclose the entire list of addresses in single quotation marks.

Note: In an SAP R/3 job, you can also use the EMAILADDR statement to email the spool file upon step completion or failure.

Example: Specify an Email List

This example emails the spool output for the SAP job to the user1@example.com and user2@example.com addresses:

```
SAPJOBNAME SAPTEST
SAPEMAILADDR 'user1@example.com,user2@example.com'
AGENT SAPHTAGENT
SAPUSER J01PRO
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPFAILUREMSG Statement—Specify Failure Message for SAP Job

The SAPFAILUREMSG statement 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 statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Job Copy (see page 164)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPFAILUREMSG '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, the scheduling manager treats this job as failed.

```
SAPJOBNAME SAPTEST
SAPFAILUREMSG 'Internal problem'
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPJOBCLASS Statement—Specify SAP Job Class

The SAPJOBCLASS statement specifies the SAP system job class the job runs under.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPJOBCLASS class
```

class

Specifies a valid SAP system job class. The value corresponds to the General data, Job class field on the Define Background Job dialog.

Limits: Up to one alphabetic character

Example: Specify an SAP System Job Class

This example runs the SAP R/3 job under the SAP system job class C:

```
SAPJOBNAME SAPTEST
SAPJOBCLASS C
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPJOBNAME Statement—Specify SAP Job Name

The SAPJOBNAME statement specifies the SAP job name.

Supported Job Types

This statement is required for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Business Warehouse InfoPackage (see page 158)
- SAP Business Warehouse Process Chain (see page 159)
- SAP Data Archiving (see page 160)
- SAP Event Monitor (see page 162)
- SAP Job Copy (see page 164)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPJOBNAME job_name
```

job_name

Specifies the SAP job name used for the SAP workload. The value corresponds to the General data, Job name field on the Define Background Job dialog.

Limits: Up to 32 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- JOBNAME is an alias of SAPJOBNAME.

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.

```
SAPJOBNAME prodrun1
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   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 single quotation marks. The quotation mark in the job name must also be duplicated.

```
SAPJOBNAME 'Today''s Work'
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPLANGUAGE Statement—Specify SAP Language for Login

The SAPLANGUAGE statement specifies the language used to log on to an SAP system.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Business Warehouse InfoPackage (see page 158)
- SAP Business Warehouse Process Chain (see page 159)
- SAP Data Archiving (see page 160)
- SAP Event Monitor (see page 162)
- SAP Job Copy (see page 164)
- SAP Process Monitor (see page 165)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPLANGUAGE language
```

language

Specifies an alphabetic character representing a valid language for SAP. The value corresponds to the SAPGUI ABAP program Variant field on the Create Step dialog.

Limits: Up to two characters; case-sensitive

Examples: EN=English, D=German, R=Russian

Note: The default is the SAP system language.

Example: Specify an SAP Language

This example uses English to log on to the SAP system:

```
SAPJOBNAME SAPTEST
SAPLANGUAGE EN
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPSUCCESSMSG Statement—Specify Success Message for SAP Job

The SAPSUCCESSMSG statement specifies a string that indicates the success of the job. If the string is found in the job's spool file, the job is considered successful.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Job Copy (see page 164)
- SAP R/3 (see page 166)

```
SAPSUCCESSMSG 'message'
```

message

Specifies a string that indicates the success of the job. If the string is found in the spool file of the SAP job, 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, the scheduling manager considers the job as successful.

```
SAPJOBNAME SAPTEST
SAPSUCCESSMSG 'Program Selections'
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
```

SAPTARGETSYSTEM Statement—Specify SAP Application Server

The SAPTARGETSYSTEM specifies the name of the SAP system application server where the SAP job runs. If you omit this statement, the SAP system selects an application server based on available resources.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Data Archiving (see page 160)
- SAP Job Copy (see page 164)
- SAP Process Monitor (see page 165)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPTARGETSYSTEM application_server
```

application_server

Specifies the host computer within the SAP system architecture that is capable of running SAP system workload.

Limits: Up to 32 valid SAP characters; case-sensitive

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER J01PROD
ABAPNAME BTCTEST
   VARIANT TEST
   SAPTARGETSYSTEM tst005
```

SAPUSER Statement—Specify the SAP System User the Job Runs Under

The SAPUSER statement specifies the SAP system user the job runs under. The SAPUSER statement overrides the SAP user specified in the connection properties file.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Business Warehouse InfoPackage (see page 158)
- SAP Business Warehouse Process Chain (see page 159)
- SAP Data Archiving (see page 160)
- SAP Event Monitor (see page 162)
- SAP Job Copy (see page 164)
- SAP Process Monitor (see page 165)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SAPUSER user_name
```

user_name

Specifies the name of the user the job runs under. The value corresponds to the SAPGUI User field on the Create Step dialog.

Limits: Up to 32 valid SAP characters; case-sensitive; it cannot contain delimiters (such as spaces)

Notes:

- If you do not specify the SAP system user using the SAPUSER statement, note the following:
  - The ABAP started by the job runs with the authority of the SAP user defined in the connection properties file for the SAP system. You can override this behavior with the STEPUSER statement.
  - The agent administrator must define the jco.client.user and jco.client.passwd properties in the connection properties file for the SAP system. For more information on the connection properties file, see the *CA Workload Automation Agent for SAP Implementation Guide*.
- If you specify the SAPUSER statement in the job definition, you must define the password using the PASSWORD command.

Example: Specify the User ID the Job Runs Under

This example runs the BTCTEST ABAP under the authority of the USERXXX01 user ID, as the STEPUSER statement is not included:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
SAPUSER USERXX01
ABAPNAME BTCTEST
   VARIANT TEST
   SAPTARGETSYSTEM tst005
```

Example: Run a Job under the Default User ID

This example runs the job under the authority of the default user ID specified in the connection properties files, as the SAPUSER and STEPUSER statements are not included:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
ABAPNAME BTCTEST
   VARIANT TEST
   SAPTARGETSYSTEM tst005
```

SAVEOUTPUT Statement—Specify Whether to Save Output from an Oracle E-Business Suite Job

The SAVEOUTPUT statement specifies whether to save the output from an Oracle E-Business Suite Single Request or Request Set job.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
SAVEOUTPUT 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.

Notes:

- Your agent administrator can specify a default setting for all Oracle E-Business Suite Request Set or Single Request jobs by setting the oa.default.saveOp parameter in the agentparm.txt file of the agent.
- The SAVEOUTPUT statement overrides the default setting specified in the oa.default.saveOp parameter in the agentparm.txt of the agent.
- 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 PROGSAVEOUTPUT statement. This statement overrides the SAVEOUTPUT statement for that particular program.
- The SAVEOUTPUT statement is ignored when Y or N is omitted.

Example: Save the Output of an Oracle E-Business Single Request Job

In the following example, the job's output is saved.

```
AGENT CYBOA
PROGRAM XXCOFI
APPLDISPLNAME 'Application Object Library'
RESPNAME 'System Administrator'
OAUSER SYSADMIN
SAVEOUTPUT Y
```

SCRIPTNAME Statement—Specify a UNIX Shell Script to Run

The SCRIPTNAME statement specifies the UNIX shell script to run.

Supported Job Type

To run shell scripts, this statement is required for the UNIX job type (see page 190).

Note: All UNIX job definitions require a SCRIPTNAME statement or a CMDNAME statement. Use the SCRIPTNAME statement to run shell scripts and the CMDNAME statement to run binary files.

Syntax

This statement has the following format:

```
SCRIPTNAME script_name
```

script_name

Specifies the full path and name of the script to run. It must be a valid executable UNIX file written in a valid scripting language.

Limits: Up to 100 characters; case-sensitive

Notes:

- 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.
- When you use the agent to run a script that calls a second script, you must provide the fully qualified path of the called 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 statement (if specified in the job definition)
  2. The first line of the script (if the SHELL statement is not specified)
  3. The oscomponent.defaultshell parameter in the agentparm.txt file (if not specified in the SHELL statement or in the script)
  4. The user default shell defined in the user profile (if not specified in one of the previous three locations)
- You must define all shells that you want the agent to use with the oscomponent.validshell parameter in the agentparm.txt file or set the oscomponent.checkvalidshell parameter to false. For more information on the oscomponent.defaultshell and oscomponent.validshell parameters, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- The SCRIPTNAME statement usually requires the full path to the script name you want to run. However, you can specify the script 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 USER statement, 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 script name is set in the PATH system environment variable for the specified user ID.
- You can specify the 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 USER statement, 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.

Example: Run a Script Using the Agent

This example runs the sort script on the UNIX_LA agent. The SHELL statement is unspecified, so the shell is picked up from either the first line of the script, the agentparm.txt file, or the user default shell in the user profile.

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test/sort
```

Example: Specify a Shell to Run the Script

This example uses the C shell to run the sort script:

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test/sort
SHELL /bin/csh
```

Example: Run a Script that is Located in a Path Set in the PATH System Environment Variable

In this example, an agent named UNIXAGENT runs a script named procscript.sh. The job runs under the user ID jsmith, which has the authority to run the script. The path to procscript.sh is set in the PATH system environment variable for jsmith on the agent computer and the agent is configured to search for paths to command and script files.

```
AGENT UNIXAGENT
SCRIPTNAME procscript.sh
USER jsmith
```

Example: Run a Script that is Located in a Path Set in a User Environment Variable

In this example, an agent named UNIXAGENT runs a script named myscript.sh. The job runs under the user ID jsmith, which has the authority to run the script. The path to myscript.sh is set in the user environment variable $MY_PATH, which is defined in the profile file for jsmith, and the agent is configured to search for paths to command and script files.

```
AGENT UNIXAGENT
SCRIPTNAME $MY_PATH/myscript.sh
USER jsmith
```

SEARCHRANGE Statement—Specify the Range to be Searched

The SEARCHRANGE statement specifies the range to be searched in a text file using a search boundary type of line, regular expression, or date and time.

Supported Job Type

This statement is required for the Text File Reading and Monitoring job type (see page 127).

Syntax

This statement has the following format:

```
SEARCHRANGE LINE|REGEX|DATETIME [FROM(lower)] [TO(upper)]
```

LINE|REGEX|DATETIME

Specifies the mode of search. Options are the following:

LINE

Searches for the text string in the line boundaries defined by the upper and lower line boundaries. If LINE is specified, the search boundaries defined by FROM and TO are numeric.

REGEX

Searches for a text string within boundaries defined by regular expressions. If REGEX is specified, the search boundaries defined by FROM and TO are regular expressions.

DATETIME

Searches for a text string within boundaries defined by a specified date and time. If DATETIME is specified, the TIMEFORMAT statement must also be defined.

FROM(*lower*)

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. When the search mode is LINE, the upper limit is 9223372036854775807.

TO(*upper*)

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:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you do not specify the FROM operand, you must specify the TO operand. The job starts searching from the first line of the text file to the upper boundary specified by the TO operand.
- If you do not specify the TO operand, you must specify the FROM operand. The job searches from the lower boundary specified by the FROM operand to the last line of the text file.
- If you specify the CONTINUOUS operand in the TEXTSTRING statement, you cannot use the TO operand in the SEARCHRANGE statement.
- When the search mode is REGEX, the search boundaries specified by the FROM and TO operands are regular expressions. 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.
- If you specify the lower boundary as a regular expression and there are no strings in the text file that match it, the agent does not search for the text string.
- If you specify the upper boundary as a regular expression and there are no strings in the text file that match it, the agent searches for the text string to the last line of the text file.
- RANGE is an alias of SEARCHRANGE.

Example: Monitor a Windows Text File for a String Using the LINE Search Mode

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.

```
AGENT WINAGENT
TEXTSTRING 'ERROR MESSAGE' EXIST
TEXTFILE 'c:\ca\logfile'
SEARCHRANGE LINE FROM(143)
```

Example: Monitor a Windows Text File for a String Using the DATETIME Search Mode

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 TIMEFORMAT statement. The job completes successfully if the transmitted string is found.

```
AGENT UNIXAGENT
TEXTFILE '/export/home/logs/transmitter.log'
TEXTSTRING transmitted EXIST
SEARCHRANGE DATETIME +
  FROM('Thu May 20 00:00:00.000 EDT 2010') +
  TO('Thu May 27 23:59:59.999 EDT 2010')
TIMEFORMAT 'EEE MMM dd HH:mm:ss.SSS zzz yyyy' TIMEPOS(12)
```

Example: Monitor a Windows Text File for a String Using the REGEX Search Mode

This example searches the agentparm.txt file in regular expression mode. The job searches the content from the lower boundary (a line that contains the word CAWA) to the upper boundary (a line that contains the word service). The job completes successfully if the archive string is found.

```
AGENT WINAGENT
TEXTFILE 'c:\program files\agentdir\agentparm.txt'
TEXTSTRING archive EXIST
SEARCHRANGE REGEX FROM(CAWA) TO(service)
```

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 TEXTSTRING statement below:

```
AGENT WINAGENT
TEXTFILE 'c:\program files\agentdir\agentparm.txt'
TEXTSTRING 'C:\\\\Program Files' EXIST
SEARCHRANGE LINE FROM(1) TO(200)
```

Example: Monitor a UNIX Text File for a String Using the DATETIME Search Mode

This example searches the /export/home/agentdir/log/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 TIMEFORMAT statement. The job completes successfully if the NTAGR6 string is found.

```
AGENT UNIXAGENT
TEXTFILE '/export/home/agentdir/log/transmitter.log'
TEXTSTRING NTAGR6 EXIST
SEARCHRANGE DATETIME+
  FROM('Thu May 20 00:00:00.000 EDT 2010')+
  TO('Thu May 27 23:59:59.999 EDT 2010')
TIMEFORMAT 'EEE MMM dd HH:mm:ss.SSS zzz yyyy' TIMEPOS(1)
```

Example: Monitor a UNIX Text File for a String at the Beginning of a Line

This example searches the /export/home/agentdir/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).

```
AGENT UNIXAGENT
TEXTFILE '/export/home/ca/agentdir/agentparm.txt'
TEXTSTRING \.0/MAIN$ EXIST
SEARCHRANGE REGEX FROM(\Aagent) TO(level=2\Z)
```

Example: Monitor a UNIX Text File for a String Using the REGEX Search Mode

This example searches for the string "# Runner plugin parameters", which is evaluated as a regular expression. The job searches from the first line that contains the word CAWA to the line that contains the word service.

```
AGENT UNIXAGENT
TEXTFILE '/export/home/agentdir/agentparm.txt'
TEXTSTRING \A\W\sRunner EXIST
SEARCHRANGE REGEX FROM(CAWA) TO(service)
```

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 a UNIX Text File for a String Using the LINE Search Mode

This example searches the test file for the text string "agent".

```
AGENT UNIXAGENT
TEXTFILE '/export/home/log/test'
TEXTSTRING agent EXIST CONTINUOUS(text)
SEARCHRANGE LINE FROM(1)
```

When the job first runs, it searches the content between line 1 and the end of the file. An alert is triggered the first time that the string is found. 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 "agent" string in the data that has already been searched. For example, suppose that the job has already searched lines 1 to 100 of the file. The file is then modified to include "agent" on line 30. During continuous monitoring, an alert is not triggered for that occurrence.

This job runs until it is completed manually.

Example: Monitor an i5/OS Text File for a String in the First 20 Lines of the File

This example searches for a text string in the DATA member of a QSYS file object on an i5/OS computer. The job searches the content between lines 1 and 20. The job completes successfully if the string is found.

```
AGENT i5AGENT
TEXTFILE /QSYS.LIB/LIBRARY.LIB/RESULTS.FILE/DATA.MBR
TEXTSTRING 'Create file failed' EXIST
SEARCHRANGE LINE FROM(1) TO(20)
```

SERVERADDR Statement—Specify a Remote Server Name

The SERVERADDR statement specifies a remote server name.

Supported Job Types

This statement is required for the following job types:

- FTP (see page 96)
- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
SERVERADDR server
```

server

Specifies the DNS name or IP address of a remote server. *serverport* is a server port number, which is mandatory for the MF_JOB type but is not required for other job types.

Limits: Up to 100 characters; case-sensitive

Example: 172.24.36.107:7010 (IPv4) or [0:0:0:0:0:FFFF:192:168:00:00]:7010 (IPv6)

Example: Download a File from a UNIX Computer

This example downloads a UNIX file from the hpsupport server with port 5222:

```
AGENT UNIXAGENT
USER lmla
SERVERADDR hpsupport
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /u1/test/ftpdata/textfile
LOCALFILENAME /export/home/test/test1
```

Example: Download a File from a UNIX Computer to a Windows Computer

This example downloads a file from a UNIX server with IP address 172.16.0.0 and port 21 to a Windows computer:

```
AGENT WINAGENT
USER ftpuser
SERVERADDR 172.16.0.0
SERVERPORT 21
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /u1/test/ftpdata/textfile
LOCALFILENAME c:\test
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5agent
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

SERVERNAME Statement—Specify a Server to Run the PeopleSoft Job

The SERVERNAME statement specifies the name of the target server that runs the PeopleSoft job.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
SERVERNAME 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 SERVERNAME statement overrides the default setting specified in the ps.default.serverName parameter in the agent's agentparm.txt.
- If the server name was not defined in the agentparm.txt file or in a SERVERNAME statement, the PeopleSoft Scheduler determines the server that will run the job.
- If the server name is not defined on the agent or using the SERVERNAME statement, 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 AEMINITEST. The server named PSNT runs the job.

```
AGENT PSAGENT
PROCESSTYPE 'Application Engine'
OUTDESTTYPE WEB
PROCESSNAME AEMINITEST
OUTDESTFORMAT PTF
PSOPRID VP1
RUNCONTROLID test
SERVERNAME PSNT
DISTRFOLDER GENERAL
```

SERVERPORT Statement—Specify a Remote Server Port

The SERVERPORT statement specifies the port number of the remote FTP server.

Supported Job Types

This statement is optional for the following job types:

- FTP (see page 96)
- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

SERVERPORT *port*

port

Specifies the port number of the remote server.

Limits: 0-65535

Default: 21

Example: Download a UNIX File

This example downloads a file from a UNIX server with IP address 172.16.0.0 and port 5222 to another UNIX computer:

```
AGENT UNIXAGENT
USER test
SERVERADDR 172.16.0.0
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME /export/home/test/ftpdata/textfile1
LOCALFILENAME /export/home/test/ftpdata/folder/textfile1
```

Example: Download a Windows File

This example downloads a file from a Windows server with IP address 172.31.255.255 and port 23 to another Windows computer:

```
AGENT WINAGENT
USER test
SERVERADDR 172.31.255.255
SERVERPORT 23
TRANSFERDIRECTION D
TRANSFERCODETYPE A
REMOTEFILENAME c:\qatest\ftpdata\textfile1
LOCALFILENAME 'c:\qatest\ftpdata\D folder\textfile1'
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5agent
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

SERVICE Statement—Specify Windows Service Name

SERVICE is an alias of SERVICENAME (see page 578).

SERVICENAME Statement—Specify the Web Service Name Within the Target Namespace (Web Service Jobs)

The SERVICENAME statement specifies the web service name within the target namespace in a Web Service job.

Supported Job Type

This statement is optional for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
SERVICENAME service
```

service

Specifies the web service name within the target namespace.

Limits: Up to 100 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- In a Web Service job, if you specify the WSDL_URL statement but not the ENDPOINT_URL statement, you must specify both the SERVICENAME and PORTNAME statements.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

SERVICENAME Statement—Specify the Name of the Windows Service to be Monitored (Windows Service Monitoring Jobs)

The SERVICENAME statement specifies the name of the Windows service to monitor.

Supported Job Type

This statement is required for the Windows Service Monitoring job type (see page 132).

Syntax

This statement has the following format:

```
SERVICENAME service_name
```

service_name

Specifies the name of the local Windows service to be monitored.

Limits: Up to 256 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- SERVICE is an alias of SERVICENAME.

Example: Monitor a Windows Service Name

This example monitors a Windows service named Log App. The job waits until the service status is CONTINUE_PENDING before it completes.

```
AGENT WINAGENT
SERVICENAME 'Log App'
STATUS CONTINUE_PENDING WAIT
```

SERVLET_URL Statement—Specify a Host URL

The SERVLET_URL statement specifies the host where the program you want to invoke resides in a HTTP job. The statement also specifies whether the agent ignores default proxy setting properties such as initialization parameters.

Supported Job Type

This statement is required for the HTTP job type (see page 42).

Syntax

This statement has the following format:

```
SERVLET_URL servlet_url [IGNORE]
```

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 program or servlet to be invoked.

Limits: Up to 1024 characters; case-sensitive

Example: http://localhost/cgi-bin/test.sh

Note: HTTP and HTTPS are supported..

IGNORE

(Optional) 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.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- If you omit the program or servlet path (action) in this statement, you must specify the ACTION statement in the job definition. If the ACTION statement is omitted from the job definition, the agent assumes that the SERVLET_URL statement specifies the full path to the program or servlet.
- If you specify the program or servlet path (action) in this statement, do not include the ACTION statement in the job definition. If the ACTION statement is included in the job definition, the agent assumes that the SERVLET_URL statement 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.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://google.com/search
PARAMETER KEYWORD(q) VALUE('ca workload automation')
AUTHORDER (BASIC, DIGEST, NTLM)
FILTER .*AE.*
```

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 user and BASIC, DIGEST, and NTLM protocols for web server authentication.

```
AGENT APPAGENT
INVOCATIONTYPE GET
SERVLET_URL http://host.example.com/protected IGNORE
CONNECTIONORIGIN host.example.com
CONNECTIONDOMAIN windows_domain
AUTHORDER (BASIC, DIGEST, NTLM)
CONNECTIONUSER user1
PROXYDOMAIN http://host.domain.proxy
PROXYHOST proxy.example.com
PROXYORIGIN http://host.origin.proxy
PROXYPORT 90
PROXYUSER user01 domain(mydomain)
```

SHELL Statement—Specify UNIX Shell to Run Script

The SHELL statement specifies the shell used to run the script named in the SCRIPTNAME statement.

A shell is a program that provides an environment for users to interact with the system. Shell programs are files that contain a series of user commands that are grouped, using logic provided by the shell, to accomplish a task.

Supported Job Type

This statement is optional for the UNIX job type (see page 190).

Syntax

This statement 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)

Notes:

- 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 statement (if specified in the job definition)
  2. The first line of the script (if the SHELL statement is not specified)
  3. The oscomponent.defaultshell parameter in the agentparm.txt file (if not specified in the SHELL statement or in the script)
  4. The user default shell defined in the user profile (if not specified in one of the previous three locations)

- You must define all shells that you want the agent to use with the oscomponent.validshell parameter in the agentparm.txt file or set the oscomponent.checkvalidshell parameter to false. For more information on the oscomponent.defaultshell and oscomponent.validshell parameters, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- Different shells provide different facilities and have different characteristics. Certain 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 on the UNIX_LA agent computer:

```
AGENT UNIX_LA
SCRIPTNAME /mfg/test/sort
SHELL /bin/csh
```

SHORTNAME Statement—Specify the Short Name of an Oracle Applications Application

SHORTNAME is an alias of APPLSHORTNAME (see page 228).

SITECMD Statement—Specify Site-specific FTP Commands

The SITECMD statement specifies the commands to execute prior to file transfer. You can use this statement to send site-specific FTP commands to FTP servers.

Supported Job Type

This statement is optional for the FTP job type (see page 96).

Syntax

This statement has the following format:

```
SITECMD command
```

command

Defines a command that is to be executed prior to file transfer.

Limits: Up to 4070 characters

Note: You can specify any number of SITECMD statements in an FTP job.

Example: Send Two FTP Commands to an FTP Server

This example sends two FTP commands to the FTP server prior to transferring a file:

```
AGENT FTPAGENT
USER user1
SERVERADDR ftp.example.com
SERVERPORT 21
TRANSFERDIRECTION D
TRANSFERCODETYPE E COMPRESS(8)
REMOTEFILENAME /pub/cazip.exe
LOCALFILENAME /tmp/bla
SITECMD site date
SITECMD site recfm=FB
```

SKIPPARMUPDATES Statement—Specify Whether to Update Job Parameters

The SKIPPARMUPDATES statement specifies whether the agent updates job parameters with data in the PS_PRCSDEFN table.

Note: We recommend that you skip parameter updates when some bind variables in the PS_PRCSDEFN table may not be suitably defined.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
SKIPPARMUPDATES YES|NO
```

YES

Specifies that the agent does not update job parameters with data in the PS_PRCSDEFN table. If you specify YES, you can use the ARGS statement to pass additional argument values.

NO

Specifies that the agent updates job parameters with data in the PS_PRCSDEFN table. This is the default.

Example: Ignore Parameters in the PS_PRCSDEFN Table and Pass Additional Parameters to the nVision Report

In this example, the job parameters are not updated with data in the PS_PRCSDEFN table. The ARGS statement passes additional parameters to the nVision report.

```
AGENT PSAGENT
PROCESSTYPE nVision-Report
PROCESSNAME NVSRUN
OUTDESTTYPE FILE
OUTDESTFORMAT XLS
OUTDESTPATH c:\test\testnv.xls
SKIPPARMUPDATES Yes
ARGS -NRNVARIABLE -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_PRCSDEFN Table

In this example, the job parameters are updated with data in the PS_PRCSDEFN table.

```
RUNCONTROLID sample
PROCESSTYPE Crystal
PROCESSNAME 'OMC3220-'
SKIPPARMUPDATES N
PSOPRID VP2
```

SNMPNODE Statement—Specify an SNMP OID, Host, Port, Version, and Other Options

The SNMPNODE statement specifies the SNMP OID (object identifier), host, port, version, and other options.

Supported Job Types

This statement is required for the following job types:

- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

- For the SNMP Trap Send job type:

  ```
  SNMPNODE oid HOST(host) [PORT(port)] VERSION(1|2|3) [INFORM]
  ```

- For the SNMP Value Get job type:

  ```
  SNMPNODE oid HOST(host) [PORT(port)] VERSION(1|2|3) [SUBTREE]
  ```

- For the SNMP Value Set job type:

  ```
  SNMPNODE oid HOST(host) [PORT(port)] VERSION(1|2|3)
  ```

oid

Specifies the SNMP OID (object identifier) in numeric or string format. In an SNMP Trap Send job, it identifies the trap that you want to send. In an SNMP Value Get or Value Set job, it identifies the variable whose value you want to retrieve or change.

Limits: Up to 256 characters; case-sensitive

Examples: agentLogLevel, .1.3.6.1.2.1.1.6.0

Note: In an SNMP Value Set job, the MIB statement is required if the OID is specified in string (non-numeric) format.

HOST(*host*)

Specifies the host name or IP address of the network device. In an SNMP Trap Send job, it identifies the computer you want to send the SNMP trap to. In an SNMP Value Get or Value Set job, it identifies the device that hosts the MIB variable.

Limits: Up to 256 characters; case-sensitive

PORT(*port*)

(Optional) Specifies the port of the network device. In an SNMP Trap Send job, it identifies the port used to send the SNMP trap. In an SNMP Value Get or Value Set job, it identifies the port used to communicate with the network device.

Limits: Up to 5 digits

Default: 161 (SNPG_JOB and SNPS_JOB); 162 (SNPE_JOB)

VERSION(1|2|3)

Specifies the SNMP version. In an SNMP Trap Send job, it identifies the version used when generating the SNMP trap. In an SNMP Value Get or Value Set job, it identifies the version used when connecting to the network device. Options include the following:

1

Specifies SNMP Version 1 (v1).

2

Specifies SNMP Version 2 (v2).

3

Specifies SNMP Version 3 (v3).

INFORM

(Optional) Indicates that the SNMP Trap Send job sends the SNMP v2/v3 INFORM message if it exists. This option is recommended for SNMP v3.

SUBTREE

(Optional) Indicates that the SNMP Value Get job walks the whole SNMP subtree starting with the OID (*oid*).

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

Example: Send an SNMP Trap to a Network Device

Suppose that you want to send the cybtrapstart trap to a network device using SNMP v3. In this example, five string parameters are passed to the trap. The host name of the network device is localhost and its port is 162. The job uses the AES privacy protocol and the SHA authentication protocol. The name of the MIB file is RFC1213-MIB.mib and the default engine ID is used. The credentials of user1 are used for authorization.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE cybtrapstart HOST(localhost) PORT(162) VERSION(3)
SNMPUSER user1
AUTHPROTOCOL SHA
PRIVPROTOCOL AES
ENGINEID
PARAMETER TYPE(snmp:string) VALUE(p1)
PARAMETER TYPE(snmp:string) VALUE(p2)
PARAMETER TYPE(snmp:string) VALUE(p3)
PARAMETER TYPE(snmp:string) VALUE(p4)
PARAMETER TYPE(snmp:string) VALUE(p5)
```

Example: Query a Network Device for the Value of an SNMP Variable

Suppose that you want to know the value of the agentVersion variable hosted by a network device. In this example, the host name of the network device is host.example.com and its port is 161. The SNMP version is v2 and the read community is public. The name of the MIB file is RFC1213-MIB.mib, which is located on the agent computer.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE agentVersion HOST(host.example.com) PORT(161) VERSION(2)
COMMUNITY public
```

Example: Query a Network Device for the Values of a Whole SNMP Subtree

Suppose that you want to retrieve the values of a whole SNMP subtree starting with the pluginManagerPluginsTable OID.

```
AGENT SNMPAGENT
MIB 'C:\SNMP\MIBs\RFC1213-MIB.mib'
SNMPNODE pluginManagerPluginsTable HOST(host.example.com) PORT(161) VERSION(2) +
  SUBTREE
COMMUNITY public
```

SNMPUSER Statement—Specify an SNMP v3 User Name

The SNMPUSER statement specifies the user name whose credentials are used for authorization in SNMP v3. This statement is ignored if your SNMP version is v1 or v2.

Supported Job Types

If your SNMP version is v3, this statement is required for the following job types:

- SNMP Trap Send (see page 179)
- SNMP Value Get (see page 181)
- SNMP Value Set (see page 183)

Syntax

This statement has the following format:

```
SNMPUSER user_name
```

user_name

Specifies the user name whose credentials are used for authentication.

Limits: Up to 128 characters; case-sensitive; it cannot contain delimiters (such as spaces)

Note: Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

Example: Specify SNMP User in an SNMP Value Set Job

Suppose that you want to set the value of the OUT456789 variable using SNMP v3. In this example, the job uses the AES privacy protocol and the MD5 authentication protocol. The OUT456789 variable belongs to the context named contxtname within the SNMP entity identified by the hexadecimal value CAB0. The credentials of user user1 are used for authorization.

```
AGENT SNMPAGENT
SNMPUSER user1
MIB 'c:\SNMP\MIBs\mymib.mib'
SNMPNODE OUT456789 HOST(host.example.com) VERSION(3)
CONTEXTNAME contxtname
CONTEXTENGINEID X'cab0'
PRIVPROTOCOL AES
AUTHPROTOCOL MD5
PARAMETER TYPE(snmp:int) VALUE(8)
```

SPNAME Statement—Specify Stored Procedure or Stored Function to Run

SPNAME is an alias of STORED_PROCEDURE (see page 597).

SPOOLRECIPIENT Statement—Specify SAP Spool Recipient

The SPOOLRECIPIENT statement specifies the recipient for the SAP spool. The recipient receives the spool upon job completion or failure.

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
SPOOLRECIPIENT recipient
```

recipient

Specifies the SAP user, SAPoffice distribution list, or email address that receives the spool.

Limits: Up to 240 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To specify multiple recipients, use multiple SPOOLRECIPIENT statements or define a distribution list in SAP and use the DLI recipient type.

Example: Specify an Email Address

This example sends the spool to the address user1@example.com upon job completion:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE MAIL
SPOOLRECIPIENT user1@example.com
ABAPNAME RSPARAM
```

Example: Specify an SAP User

This example sends the spool to the SAP user J01Prod upon job completion:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE SU
SPOOLRECIPIENT J01Prod
ABAPNAME BTCTEST
```

Example: Specify an SAPoffice Distribution List

This example sends the spool to the payroll distribution list upon job completion:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
RECIPIENTTYPE DLI
SPOOLRECIPIENT payroll
ABAPNAME BTCTEST
```

SQL Statement—Specify SQL to Run Against a Database Table

The SQL statement specifies an SQL statement to run against a database table.

Supported Job Type

This statement is required for the SQL job type (see page 81).

Syntax

This statement has the following format:

SQL *sql_statement*

sql_statement

Specifies the SQL statement to run against a database table.

Limits: Up to 4078 characters; case-sensitive

Example: 'SELECT store FROM store_info WHERE sales > 1000'

Note: The value can contain any SQL statement including SELECT, DELETE, UPDATE or INSERT.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- To use the SQL statement, you must know how to construct an SQL query.
- SQL_COMMAND is an alias of SQL.

Example: Run an SQL Query

In the following SQL job, the agent CYBDB1 runs the SQL query 'SELECT store FROM store_info WHERE sales > 1000':

```
AGENT CYBDB1
SQL 'SELECT store FROM store_info WHERE sales > 1000'
```

SQL_COMMAND Statement—Specify SQL to Run Against a Database Table

SQL_COMMAND is an alias of SQL (see page 591).

STARTMODE Statement—Specify Startmode for Readied Job

The STARTMODE statement specifies the action to take when the job is readied on the SAP system.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP Job Copy (see page 164)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
STARTMODE N|I|A
```

N

Does not release the job.

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.

Example: Release the Job Immediately

This example runs the job immediately assuming free background processing is available:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
STARTMODE I
ABAPNAME BTCTEST
   VARIANT TEST
   SAPTARGETSYSTEM tst005
```

STATUS Statement—Specify the Status of the Process, Service, or IP Address to Monitor

The STATUS statement specifies the status of the process, service, or IP address to monitor. The job checks whether the status of the process, service, or IP address matches the status specified by this statement.

Supported Job Types

This statement is required for the following job types:

- IP Monitoring (see page 125)
- Process Monitoring (see page 126)
- Windows Service Monitoring (see page 132)

Syntax

This statement has the following format:

- For the IP Monitoring and Process Monitoring job types:

  ```
  STATUS RUNNING|STOPPED
         NOW|WAIT
  ```

- For the Windows Service Monitoring job type:

  ```
  STATUS RUNNING|STOPPED|CONTINUE_PENDING|PAUSE_PENDING|PAUSED|START_PENDING|
          STOP_PENDING|EXISTS|NOTEXISTS
         NOW|WAIT
  ```

RUNNING

Specifies that the job monitors the process, service, or IP address for a running status.

STOPPED

Specifies that the job monitors the process, service, or IP address 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.

NOW

Returns the status of the process, service, or IP address immediately.

Note: If the current status matches the status type, the job completes successfully; otherwise, the job fails.

WAIT

Monitors the process, service, or IP address continuously until it reaches the specified status.

Note: On UNIX and i5/OS, you can only specify the RUNNING, STOPPED, NOW, and WAIT operands.

Example: Monitor a Stopped Process

This example monitors the Bayuser process. The job checks the process status immediately and completes successfully if the process is stopped. If the process is running, the job fails.

```
AGENT SYSAGENT
PROCESS Bayuser
STATUS STOPPED NOW
```

Example: Monitor a Running Process

This example monitors the nlnotes process. 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.

```
AGENT SYSAGENT
PROCESS nlnotes
STATUS RUNNING WAIT
```

Example: Monitor the Agent Process on Windows Using a Full Path

Suppose that the WINAGENT computer runs multiple agents. To monitor the cybAgent.exe process, the full path to that process is specified. The job checks the process status immediately and completes successfully if the process is running. If the process is not running, the job fails. The process name is enclosed in single quotes because it contains a space.

```
AGENT WINAGENT
PROCESS 'c:\Program files\agentdir\cybAgent.exe'
STATUS RUNNING NOW
```

Note: For a UNIX example, substitute a UNIX path in the PROCESS statement.

Example: Monitor an IP Address for a Running Status

This example monitors a device at IP address 10.1.2.20 and port 22. When the job runs, it immediately checks if the device is running. If the device is running, the job completes. If the device is not running, the job fails.

```
AGENT SYSAGENT
IPADDRESS 10.1.2.20
IPPORT 22
STATUS RUNNING NOW
```

Example: Monitor a Running Service

This example monitors a Windows service named Proc Server. If the service is running, the job completes successfully. If the service is not running, the job continues monitoring it until the service starts running.

```
AGENT WINAGENT
SERVICENAME 'Proc Server'
STATUS RUNNING WAIT
```

STEPUSER Statement—Specify the SAP System User the ABAP Runs Under

The STEPUSER statement specifies the user under whose authorization the ABAP program runs.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
STEPUSER user_name
```

user_name

Specifies the name of the user under whose authorization the ABAP program runs.

Limits: Up to 16 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the STEPUSER statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the STEPUSER statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.
- The STEPUSER statement overrides the SAPUSER statement.

Example: Specify the User for an ABAP Program

This example runs the BTCTEST ABAP under the authority of the user14 user ID:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
STARTMODE I
ABAPNAME BTCTEST
   STEPUSER user14
   VARIANT TEST
   SAPTARGETSYSTEM tst005
```

STORED_PROCEDURE Statement—Specify Stored Procedure or Stored Function to Run

The STORED_PROCEDURE statement specifies the database stored procedure to run. If you are using an Oracle or SQL Server database, you can also use the STORED_PROCEDURE statement to specify a stored function.

Supported Job Type

This statement is required for the Database Stored Procedure job type (see page 71).

Syntax

This statement has the following format:

Type

```
STORED_PROCEDURE 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 100 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- The agent does not support stored functions (also named user-defined functions) on DB2.
- SPNAME is an alias of STORED_PROCEDURE.

Example: Run a Stored Procedure

The following example runs the stored procedure UPDATE_INV:

```
AGENT CYBDB1
STORED_PROCEDURE UPDATE_INV
```

SUCCESSMSG Statement—Specify Success Message for ABAP

The SUCCESSMSG statement specifies a string that indicates the success of the SAP step (ABAP). If the string matches the SAP ABAP output for the step, the step is considered successful.

Supported Job Type

This statement is optional for the SAP R/3 job type (see page 166).

Syntax

This statement has the following format:

```
SUCCESSMSG 'message'
```

message

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:

- Use the SUCCESSMSG statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the SUCCESSMSG statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Use a Success Message to Determine an ABAP's Success

This example defines the success message, Program Selections. If the string is found in the ABAP spool output, the scheduling manager considers the BTCTEST ABAP as successful.

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
STARTMODE I
ABAPNAME BTCTEST
   STEPUSER user14
   VARIANT TEST
   SUCCESSMSG 'Program Selections'
```

TABLE_NAME Statement—Specify the Name of the Table to Monitor

The TABLE_NAME statement specifies the name of the database table to monitor for the changes specified in the Database Monitor or Database Trigger job.

Supported Job Types

This statement is required for the following job types:

- Database Monitor (see page 69)
- Database Trigger (see page 76)

Syntax

This statement has the following format:

```
TABLE_NAME table [CONTINUOUS(alertid)]
```

table

Specifies the name of the database table to monitor for changes.

Limits: Up to 100 characters; case-sensitive

CONTINUOUS(*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified database monitoring conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the specified database monitoring conditions occur.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- TABLENAME is an alias of TABLE_NAME.

Example: Monitor a Table for an Update

In the following example, the table Inventory_List is monitored for an update. When an update occurs, the job completes.

```
AGENT CYBDB1
TABLE_NAME Inventory_List
TRIG_TYPE UPDATE
```

Example: Monitor a Table Continuously for the Insertion of Rows

In the following example, the table Inventory_List is continuously monitored for the insertion of table rows. When a row is inserted, the scheduling manager triggers the predefined alert INC. After the alert is triggered, monitoring resumes. To end the job, you must complete it manually.

```
AGENT CYBDB1
TABLE_NAME Inventory_List CONTINUOUS(INC)
TRIG_TYPE INSERT
```

Example: Monitor a Table Continuously for Updates

In the following example, the table Inventory_List is continuously monitored for updates. When an update occurs, if the number of units of product A has fallen below 100000, the scheduling manager triggers the predefined alert LOW. After the alert is triggered, monitoring resumes. To end the job, you must complete it manually.

```
AGENT CYBDB1
TABLE_NAME Inventory_List CONTINUOUS(LOW)
TRIG_TYPE UPDATE
TRIG_COND 'new.ProductA < 100000'
```

TABLENAME Statement—Specify the Name of the Table to Monitor

TABLENAME is an alias of TABLE_NAME (see page 599).

TARGETNAMESPACE Statement—Specify a Target Namespace

The TARGETNAMESPACE statement specifies the target namespace in a Web Service job.

Supported Job Type

This statement is required for the Web Service job type (see page 211).

Syntax

This statement has the following format:

```
TARGETNAMESPACE 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

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```

TARGETOSTYPE Statement—Specify the Remote Operating System Type

The TARGETOSTYPE statement specifies the remote operating system type in a secure file transfer. The remote operating system type is used to determine the path separator on the remote system.

Supported Job Types

This statement is optional for the following job types:

- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
TARGETOSTYPE UNIX|Windows|Open VMS|Tandem|OS400|MVS
```

UNIX

Specifies that the remote operating system is UNIX. This is the default.

Windows

Specifies that the remote operating system is Windows.

Open VMS

Specifies that the remote operating system is OpenVMS.

Tandem

Specifies that the remote operating system is Tandem.

OS400

Specifies that the remote operating system is i5/OS.

MVS

Specifies that the remote operating system is z/OS.

Example: Specify the Remote Operating System Type

This example downloads the file_size8 file from the /ul/test/ftpdata directory on the simon server using the Secure Copy Protocol (SCP). After the file is downloaded, the agent computer contains a file named scp_file_size8_local_user in the /u1/causer/data directory. The owner of the file is test. The remote operating system type is UNIX.

```
AGENT UNIXAGENT
LOCALNAME /ul/causer/data/scp_file_size8_local_user
REMOTENAME file_size8
REMOTEDIR /u1/test/ftpdata
SERVERADDR simon
SERVERPORT 22
TRANSFERDIRECTION DOWNLOAD
USER causer
LOCALUSER test
TARGETOSTYPE UNIX
```

TEMPLATELANG Statement—Specify the Template Language for a Single Request or Request Set

The TEMPLATELANG statement specifies the template language for a single request or request set.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

`TEMPLATELANG` *`template_language`*

template_language

Specifies the template language for a single request or request set. In Oracle Applications, the template language is specified as a request definition option and is found in the Template Language column of the Upon Completion dialog.

Limits: Up to ten characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.templateLanguage parameter in the agent's agentparm.txt file.
- The TEMPLATELANG statement overrides the default setting specified in the oa.default.templateLanguage parameter in the agent's agentparm.txt.

Example: Specify the Template Language

Suppose that you want to define a Single Request job to run a single request program on the CYBOA agent. The single request belongs to the FND application in Oracle Applications and runs the program FNDSCURS. The job runs under the SYSADMIN user with System Administrator responsibility and includes layout template options.

```
AGENT CYBOA
PROGRAM FNDSCURS
APPLSHORTNAME FND
RESPNAME 'System Administrator'
OAUSER SYSADMIN
OUTPUTFORMAT PDF
TEMPLATELANG EN
TEMPLATETERR US
```

TEMPLATETERR Statement—Specify the Template Territory for a Single Request or Request Set

The TEMPLATETERR statement specifies the template territory for a single request or request set.

Supported Job Types

This statement is optional for the following job types:

- Oracle E-Business Suite Request Set (see page 137)
- Oracle E-Business Suite Single Request (see page 142)

Syntax

This statement has the following format:

```
TEMPLATETERR template_territory
```

template_territory

Specifies the template territory for a single request or request set. In Oracle Applications, the template territory is specified as a request definition option and is found in the For Language column of the Upon Completion dialog.

Limits: Up to ten characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.templateTerritory parameter in the agent's agentparm.txt file.
- The TEMPLATETERR statement overrides the default setting specified in the oa.default.templateTerritory parameter in the agent's agentparm.txt.

Example: Specify the Template Territory

Suppose that you want to define a Single Request job to run a single request program on the CYBOA agent. The single request belongs to the FND application in Oracle Applications and runs the program FNDSCURS. The job runs under the SYSADMIN user with System Administrator responsibility and includes layout template options.

```
AGENT CYBOA
PROGRAM FNDSCURS
APPLSHORTNAME FND
RESPNAME 'System Administrator'
OAUSER SYSADMIN
OUTPUTFORMAT PDF
TEMPLATELANG EN
TEMPLATETERR US
```

TEXTFILE Statement—Specify a Text File Name and Location

The TEXTFILE statement specifies the name and location (path) of the text file to search.

Supported Job Type

This statement is required for the Text File Reading and Monitoring job type (see page 127).

Syntax

This statement has the following format:

```
TEXTFILE file_name
```

file_name

Specifies the path to and name of the text file to search.

Limits: Up to 256 characters; case-sensitive

i5/OS:

- To specify a file in the root file system, use UNIX path and file formats.
- To specify a FILE object in QSYS in path format, use one of the following formats:

  /QSYS.LIB/*library*.LIB/*object*.FILE

  /QSYS.LIB/*library*.LIB/*object*.FILE/*member*.MBR
- To specify an object in QSYS in i5/OS standard format, use one of the following formats:

  library/*object*/*FILE

  library/*object*/*FILE(*member*)

library

Specifies the name of the library that contains the object. The value must be a valid i5/OS library name.

Limits: Up to 10 characters

Notes:

- You can specify *ALL to match any name.
- You can specify a generic name.

object

Specifies the name of the object. The value must be a valid i5/OS object name.

Limits: Up to 10 characters

Notes:

- You can specify *ALL to match any name.
- You can specify a generic name.

member

Specifies a member in the *FILE object. The value must be a valid i5/OS member name.

Limits: Up to 10 characters

Notes:

- You can specify *FIRST.
- You can specify *ALL to match any name.

Note: If you do not specify a *member*, the agent searches the entire file object.

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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 single quotation marks because the path contains a space. The job searches between lines 1234 and 1876. The job completes successfully if the string is found.

```
AGENT WINAGENT
TEXTSTRING 'ERROR MESSAGE' EXIST
TEXTFILE 'c:\program files\agent\log'
SEARCHRANGE LINE FROM(1234) TO(1876)
```

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 TIMEFORMAT statement. The job completes successfully if the transmitted string is found.

```
AGENT UNIXAGENT
TEXTFILE '/export/home/logs/transmitter.log'
TEXTSTRING transmitted EXIST
SEARCHRANGE DATETIME +
  FROM('Thu May 20 00:00:00.000 EDT 2010') +
  TO('Thu May 27 23:59:59.999 EDT 2010')
TIMEFORMAT 'EEE MMM dd HH:mm:ss.SSS zzz yyyy' TIMEPOS(12)
```

Example: Specify a Text File Name on i5/OS

This example searches for a text string in the DATA member of a QSYS file object on an i5/OS computer. The job searches the content between lines 1 and 20. The job completes successfully if the string is found.

```
AGENT I5AGENT
TEXTFILE /QSYS.LIB/LIBRARY.LIB/RESULTS.FILE/DATA.MBR
TEXTSTRING 'Create file failed' EXIST
SEARCHRANGE LINE FROM(1) TO(20)
```

TEXTSTRING Statement—Specify a Text String to Search For

The TEXTSTRING statement specifies the text to search for.

Supported Job Type

This statement is required for the Text File Reading and Monitoring job type (see page 127).

Syntax

This statement has the following format:

```
TEXTSTRING textstring [EXIST|NOTEXIST]
[CONTINUOUS(alertid)]
```

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

EXIST|NOTEXIST

(Optional) Specifies whether the job monitors if a text string exists or does not exist.

EXIST

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.

NOTEXIST

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 NOTEXIST setting, you must specify NOW in the WAITMODE statement.

CONTINUOUS (*alertid*)

(Optional) Specifies the identifier of an alert to be triggered when the specified text monitoring conditions occur. Defines the job as continuous. To end continuous monitoring, you must complete the job manually or cancel it. If you do not specify this operand, the job completes when the job finds the first match to the text string.

alertid

Specifies the alert identifier to be triggered.

Limits: 1-8 alphanumeric characters; the first character must be alphabetic

Note: The alert must have been previously defined to the scheduling manager. Not all scheduling managers support the CONTINUOUS operand.

Notes:

- 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 use the NOTEXIST operand in this statement with the CONTINUOUS operand.
- You cannot use the TO operand in the SEARCHRANGE statement with the CONTINUOUS operand.
- You cannot use the WAITMODE statement with the CONTINUOUS operand in the TEXTSTRING statement, they are mutually exclusive.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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: 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).

```
AGENT UNIXAGENT
TEXTFILE '/export/home/cybermation/agentdir/agentparm.txt'
TEXTSTRING '=jars/\*\.\*\Z'
SEARCHRANGE LINE FROM(1) TO(110)
```

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 operand REGEX in the SEARCHRANGE statement.

```
AGENT UNIXAGENT
TEXTFILE '/export/home/cybermation/agentdir/agentparm.txt'
TEXTSTRING '^\w{4,10}\.' Exist
SEARCHRANGE REGEX FROM(\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 following syntax:

```
TEXTSTRING '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: Continuously Monitor a File for a String

This example searches the text file DATA SOURCE NAME for the text string EVENT ""COMPUTER() .

```
AGENT WINAGENT
USER CAUSER
JOBCLASS BA
TEXTSTRING 'EVENT ""COMPUTER() ' EXIST CONTINUOUS(A123)
TEXTFILE 'DATA SOURCE NAME'
SEARCHRANGE LINE FROM(123)
```

When the job first runs, it searches the content between line 123 and the end of the file. An alert is triggered the first time that the string is found. 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 text string in the data that has already been searched. For example, suppose that the job has already searched lines 123 to 200 of the file. The file is then modified to include the text string on line 150. During continuous monitoring, an alert is not triggered for that occurrence.

This job runs until it is completed manually.

TIMEFORMAT Statement—Define a Time Format

The TIMEFORMAT statement defines the date and time pattern to use when searching a text file with a time stamp.

Supported Job Type

If the DATETIME operand is specified in a SEARCHRANGE statement, this statement is required for the Text File Reading and Monitoring job type (see page 127).

Syntax

This statement has the following format:

```
TIMEFORMAT timeformat [TIMEPOS(timepos)]
```

timeformat

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

TIMEPOS(*timepos*)

Specifies the first column of the time stamp in the log file.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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
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	

Time Format Pattern Characters			
Character	**Definition**	**Type**	**Example**
''	Single quote	Literal	'

- The following table displays sample time format patterns for July 10, 2010 at 12:08 p.m. Eastern Standard Time:

Format Pattern	Result
"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
"yyyyy.MMMMM.dd GGG hh:mm aaa"	2010.July.10 AD 12:08 PM

Example: Define a Time Format

In this example, the TIMEFORMAT statement specifies the time pattern format that is applied to the values specified by the FROM and TO operands of the SEARCHRANGE statement. The time values are enclosed in single quotation marks because they contain spaces.

The values for the TIMEFORMAT statement 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.

```
AGENT SYSAGENT
TEXTFILE '/export/home/cybermation/agentdir/log/transmitter.log'
TEXTSTRING NTAGR6 EXIST
SEARCHRANGE DATETIME+
    FROM('08/17/2010 00:00:00.000 EST-0500')+
    TO('08/25/2010 23:59:59.999 EST-0500')
TIMEFORMAT 'MM/dd/yyyy HH:mm:ss.SSS zZ' TIMEPOS(1)
```

TIMEOUT Statement—Specify the Ping Timeout

The TIMEOUT statement specifies how long the agent attempts to ping the computer after the Wake on LAN (WOL) signal is sent. You can use this statement, together with the WAKEHOST and WAKEPORTS statements, to confirm that the computer is awake.

Supported Job Type

This statement is optional for the Wake on LAN job type (see page 206).

Syntax

This statement has the following format:

```
TIMEOUT timeout
```

timeout

Specifies the timeout for the ping in seconds.

Limits: Up to 5 digits

Default: 120 seconds

Note: The agent attempts to connect to each of the ports specified in the WAKEPORTS statement (or the default ports if the WAKEPORTS statement is not specified) before measuring the time for the timeout.

Example: Broadcast the WOL Signal and Ping a Port

This example broadcasts the WOL signal to the server identified by the 172.16.00 IP address and the 00-1E-4F-C1-0F-FE MAC address. After the WOL signal is sent, the agent pings port 7 of the host computer to ensure it is available. If port 7 is available, the job completes successfully; otherwise, it fails. The job uses the default timeout for the ping.

```
AGENT AGENTNME
WAKEHOST host
WAKEPORTS 7
BROADCAST 172.16.0.0
MAC 00-1E-4F-C1-0F-FE
```

TIMEZONE Statement—Specify a Time Zone

The TIMEZONE statement specifies a different time zone for the PeopleSoft report being run.

Supported Job Type

This statement is optional for the PeopleSoft job type (see page 146).

Syntax

This statement has the following format:

```
TIMEZONE 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 AEMINITEST. The job runs in Central time in the Continental United States.

```
AGENT PSAGENT
PROCESSTYPE 'Application Engine'
OUTDESTTYPE WEB
PROCESSNAME AEMINITEST
OUTDESTFORMAT PTF
PSOPRID VP1
RUNCONTROLID test
TIMEZONE CST
DISTRFOLDER GENERAL
```

TOPIC Statement—Specify Whether to Publish or Subscribe to a Topic or Queue

The TOPIC statement 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.

Supported Job Types

This statement is required for the following job types:

- JMS Publish (see page 47)
- JMS Subscribe (see page 49)

Syntax

This statement has the following format:

```
TOPIC Y|N
```

Y

Sends or publishes messages to a topic.

N

Sends or publishes messages to a queue.

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 ID to gain access to the connection factory named ConnectionFactory.

```
AGENT APPAGENT
INITIAL_CONTEXT com.ibm.websphere.naming.WsnInitialContextFactory
LOCATION iiop://172.24.0.0:2809
CONNECTION_FACTORY ConnectionFactory
DESTNAME Queue
TOPIC N
MSGCLASS String
JNDIUSER cyberuser
PARAMETER TYPE(java.lang.String) VALUE('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.

TRANSFERCODETYPE Statement—Specify the FTP Format

The TRANSFERCODETYPE statement specifies an ASCII, binary, auto-detect, or EBCDIC transfer, FTP communication type (SSL or regular), and compression level in an FTP job.

Supported Job Type

This statement is required for the FTP job type (see page 96).

Syntax

This statement has the following format:

```
TRANSFERCODETYPE [A|B|E|U]
                 [SSL(YES|NO)]
                 [COMPRESS(0|1|2|3|4|5|6|7|8|9)]
```

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.

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.

U

Indicates an auto-detect transfer.

Note: You can specify auto detect for uploads only.

i5/OS: If the source file to be uploaded is EBCDIC, but the target does not support EBCDIC transfers, the job fails.

SSL(YES|NO)

Specifies whether to transfer the data with Secure Sockets Layer (SSL) communication or regular communication.

YES

Transfers the data using SSL FTP.

NO

Transfers the data using regular FTP.

Note: If the SSL operand is not specified, 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).

COMPRESS(0|1|2|3|4|5|6|7|8|9)

Specifies the data compression level.

Limits: 0-9 (0 is no data compression and 9 is the highest data compression)

Default: 0

Note: If the COMPRESS operand is not specified, 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).

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 SSL operand 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 SSL operand to FALSE. Otherwise, the job will fail.
- 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 COMPRESS operand overrides the default compression level specified in the agent's agentparm.txt.
- FTPFORMAT is an alias of TRANSFERCODETYPE.

Example: Download a Binary File on UNIX

The FTP job in this example uses a binary transfer.

```
AGENT UNIXAGENT
USER test
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE B
REMOTEFILENAME /u1/test/ftpdata/WAmgr
LOCALFILENAME /export/home/test/ftpdata/transf.bin
```

Example: Download a Binary File on Windows

The FTP job in this example uses a binary transfer.

```
AGENT WINAGENT
USER test
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE B
REMOTEFILENAME /u1/qatest/ftpdata/WAmgr
LOCALFILENAME c:\qatest\ftpdata\transf.bin
```

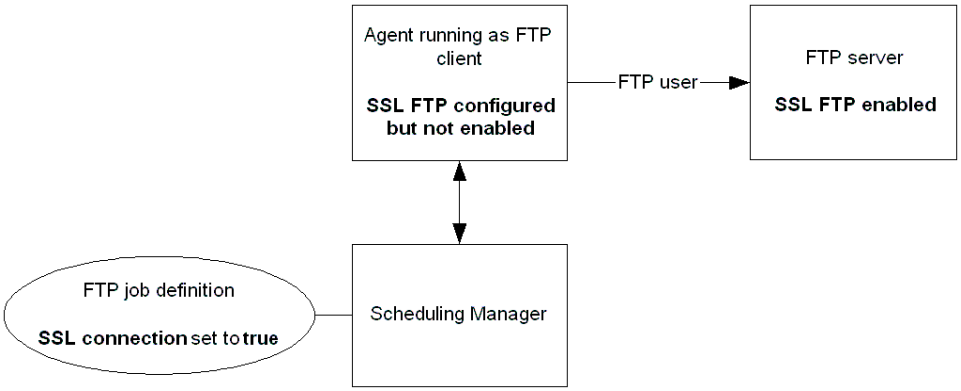
Example: Upload an ASCII-encoded File in the Root File System from an i5/OS System to a UNIX System

This example uploads an ASCII file named textfile in the root file system from an i5/OS system and copies it to a UNIX system:

```
AGENT I5AGENT
USER test
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION UPLOAD
TRANSFERCODETYPE A
LOCALFILENAME /home/ftp/textfile
REMOTEFILENAME /u1/test/ftpdata/textfile
```

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.

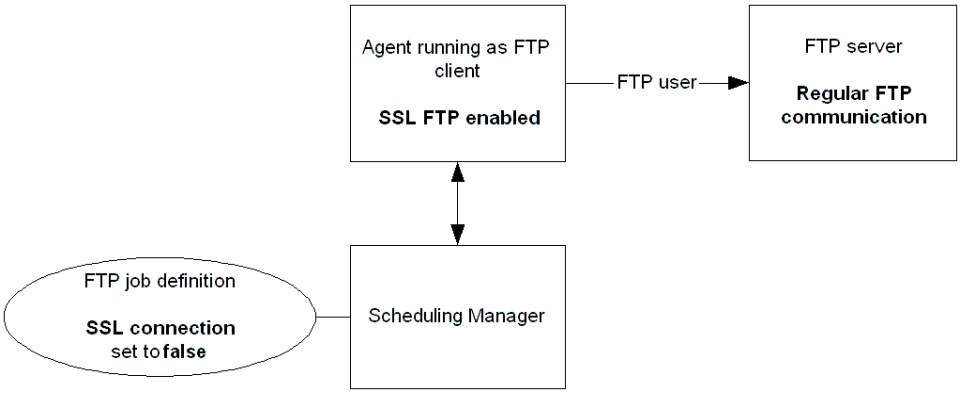


The following FTP job uploads the filename.txt file from the agent FTP client to the c:\uploaded_files directory on a remote Windows server. The SSL operand in the TRANSFERCODETYPE statement is set to YES to transfer the file securely. Although the agent FTP client does not have SSL FTP enabled, the 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).

```
AGENT WINAGENT
USER user1
SERVERADDR winserver
SERVERPORT 21
TRANSFERDIRECTION U
TRANSFERCODETYPE U SSL(YES)
REMOTEFILENAME c:\uploaded_files\filename.txt
LOCALFILENAME 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 c:\downloaded_files directory on the local computer. Because the FTP server does not support SSL FTP, the SSL operand in the TRANSFERCODETYPE statement is set to NO so the job does not fail.

```
AGENT WINAGENT
USER user1
SERVERADDR hpunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A SSL(NO)
REMOTEFILENAME /files_to_download/filename.txt
LOCALFILENAME c:\downloaded_files\filename.txt
```

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.

```
AGENT I5AGENT
USER test
SERVERADDR i5agent
SERVERPORT 5222
TRANSFERDIRECTION DOWNLOAD
TRANSFERCODETYPE E
LOCALFILENAME /QSYS.LIB/FTPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
REMOTEFILENAME /QSYS.LIB/DATALIB.LIB/DATAFILE.FILE/DATA.MBR
```

Example: Compress a File

In this example, the local computer has an agent running as an FTP client. The remote computer 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.

```
AGENT WINAGENT
USER user1
SERVERADDR aixunix
SERVERPORT 5222
TRANSFERDIRECTION D
TRANSFERCODETYPE A COMPRESS(3)
REMOTEFILENAME /files_to_download/largefile.txt
LOCALFILENAME c:\downloaded_files\largefile.txt
```

TRANSFERDIRECTION Statement—Specify the File Transfer Direction

The TRANSFERDIRECTION statement specifies the file transfer direction between the agent computer and the remote server.

Supported Job Types

This statement is required for the following job types:

- FTP (see page 96)
- Secure Copy (see page 174)
- Secure FTP (see page 175)

Syntax

This statement has the following format:

```
TRANSFERDIRECTION U|UPLOAD|
                  D|DOWNLOAD
```

U|UPLOAD

Transfers files from the agent computer to the remote server.

D|DOWNLOAD

Transfers files from the remote server to the agent computer.

Note: DIRECTION is an alias of TRANSFERDIRECTION.

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.

```
AGENT UNIXAGENT
USER test
SERVERADDR hprsupp
SERVERPORT 5222
TRANSFERDIRECTION U
TRANSFERCODETYPE A
REMOTEFILENAME /tmp
LOCALFILENAME /export/home/test/ftpdata/text*
```

TRIG_COND Statement—Specify a Condition to Monitor

The TRIG_COND statement specifies a condition to monitor the database for.

Supported Job Type

This statement is optional for the Database Trigger job type (see page 76).

Syntax

This statement has the following format:

```
TRIG_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.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- TRIGCOND is an alias of TRIG_COND.

Example: Specify Trigger Condition for Microsoft SQL Server

In the following example, a Database Trigger job monitors the SALES table for changes to the ord_date and qty columns. The job completes only when both columns have changed.

```
AGENT DB_AGENT
USER sa
DB_URL 'jdbc:sqlserver://myhost:1433;DatabaseName=pubs'
TRIG_TYPE UPDATE
TABLE_NAME SALES
TRIG_COND 'UPDATE(ord_date) and UPDATE(qty)'
```

Example: Specify Trigger Condition for IBM DB2

In the following example, a Database Trigger job monitors the STAFF table for changes to the DEPT and JOB columns. The job completes once DEPT or JOB is updated.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
TABLE_NAME STAFF
TRIG_TYPE 'UPDATE of DEPT, JOB'
```

Example: Specify Trigger Condition for Oracle

In the following example, the table Inventory_List is monitored continuously for updates. When the table is updated, if the number of units of productA has fallen below 1000, the scheduling manager triggers the predefined alert ALOW.

```
AGENT CYBDB1
TABLE_NAME Inventory_List CONTINUOUS(ALOW)
TRIG_TYPE UPDATE
TRIG_COND 'new.ProductA < 1000'
```

TRIG_TYPE Statement—Specify the Type of Database Change to Monitor For

The TRIG_TYPE statement specifies the type of database change to monitor for.

Supported Job Type

This statement is required for the Database Trigger job type (see page 76).

Syntax

This statement has the following formats:

```
TRIG_TYPE INSERT
TRIG_TYPE DELETE
TRIG_TYPE UPDATE
```

INSERT

Monitors for the insertion of a row in the database table.

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.

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- You can specify multiple trigger types for Oracle and Microsoft SQL Server:
  - For Oracle, separate the trigger types using "OR", for example, TRIG_TYPE 'INSERT OR UPDATE'.
  - For Microsoft SQL Server, separate the trigger types using a comma, for example, TRIG_TYPE 'INSERT, UPDATE'.
- For IBM DB2, you can specify multiple column names with the UPDATE operand to monitor for updates to those columns. For example, 'UPDATE of DEPT, JOB' monitors for updates to the DEPT and JOB columns.
- 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.
- TRIGTYPE is an alias of TRIG_TYPE.

Example: Specify Trigger Type for MS SQL Server

In the following example, a Database Trigger job monitors the stores table for an added row or a deleted row. When a row is either added or deleted, the job completes.

```
AGENT CYBDB
TABLE_NAME stores
TRIG_TYPE 'INSERT, DELETE'
```

Example: Specify Trigger Type for IBM DB2

In the following example, a Database Trigger job monitors the STAFF table for changes to the DEPT and JOB columns. The job completes once DEPT or JOB is updated.

```
AGENT DB_AGENT
USER entadm
DB_URL jdbc:db2://10.1.4.131:50000/SAMPLE
TABLE_NAME STAFF
TRIG_TYPE 'UPDATE of DEPT, JOB'
```

Example: Specify Trigger Type for Oracle

In the following example, the table Inventory_List is monitored continuously for the insertion of a row in the table. When a row is inserted, the scheduling manager triggers the predefined alert INC.

```
AGENT CYBDB1
TABLE_NAME Inventory_List CONTINUOUS(INC)
TRIG_TYPE INSERT
```

TRIGCOND Statement—Specify a Condition to Monitor

TRIGCOND is an alias of TRIG_COND (see page 626).

TRIGTYPE Statement—Specify the Type of Database Change to Monitor

TRIGTYPE is an alias of TRIG_TYPE (see page 628).

TYPESFILTER Statement—Specify the Notification Types to Subscribe to

The TYPESFILTER statement specifies a list of notification types to subscribe to in a JMX-MBean Subscribe job. A JMX-MBean Subscribe job can monitor an MBean for a single notification or monitor continuously for notifications. You can use the TYPESFILTER statement to filter the notifications by notification type.

Supported Job Type

This statement is optional for the JMX-MBean Subscribe job type (see page 58).

Syntax

This statement has the following format:

```
TYPESFILTER type[,type...]
```

type

Identifies a notification type to monitor for.

Limits: Case-sensitive

Example: jmx.attribute.change (sends a notification every time an attribute changes)

Notes:

- Enclose value that contain delimiters (such as spaces) in single quotation marks.
- In a list, enclose values that contain commas in single quotes.
- The entire value can be to 1024 characters.
- In a JMX-MBean Subscribe job, you can specify the ATTRIBUTESFILTER statement or the TYPESFILTER statement, but not both. If you do not specify either statement, the job will subscribe to all notifications for the MBean.

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 named CHGA is sent.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config' CONTINUOUS(CHGA)
TYPESFILTER jmx.attribute.change
```

ULIMIT Statement—Specify UNIX Resource Limits

The ULIMIT statement modifies resource limits on the agent computer for the duration of the job. For example, you can define a job that modifies the maximum core file size and CPU times on the UNIX computer.

Supported Job Type

This statement is optional for the UNIX job type (see page 190).

Syntax

This statement has the following format:

```
ULIMIT TYPE(C|D|F|M|N|S|T) SOFT(soft_value) [HARD(hard_value)]
```

TYPE(C|D|F|M|N|S|T)

Specifies the resource type. Options are the following:

- C—Specifies the core file size in kilobytes (KB).
- D—Specifies the data segment size in kilobytes (KB).
- F—Specifies the maximum file size in kilobytes (KB).
- 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(***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(***hard_value***)

(Optional) 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

Default: soft limit value

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:

- The entire value can be up to 4096 characters.
- To specify multiple ulimit values, define a separate ULIMIT statement for each value.
- 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 logon scripts.

Example: Specify Multiple Resource Limits

This example runs the procrun.sh script on the UNIXAGT agent. The job modifies the following resource limits on the UNIX computer:

- The core file size limit is 100 KB (soft limit). The size can increase to 200 KB (hard limit).
- The stack size limit is 250 KB (soft limit). The size can increase to 300 KB (hard limit).
- The CPU time limit is 1000 seconds (soft limit). The time can increase to 4000 seconds (hard limit).
- The process virtual size limit is 3332 KB (soft limit). The size can increase to an unlimited value.

```
AGENT UNIXAGT
SCRIPTNAME /ul/procrun.sh
ULIMIT TYPE(C) SOFT(100) HARD(200)
ULIMIT TYPE(S) SOFT(250) HARD(300)
ULIMIT TYPE(T) SOFT(1000) HARD(4000)
ULIMIT TYPE(M) SOFT(3332) HARD(unlimited)
```

URL Statement—Specify the URL of the JMX Server

The URL statement specifies the URL to connect to the JMX server in a JMX job.

Supported Job Types

This statement is required for the following job types:

- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Create Instance (see page 55)
- JMX-MBean Operation (see page 56)
- JMX-MBean Remove Instance (see page 57)
- JMX-MBean Subscribe (see page 58)

Syntax

This statement 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 1024 characters; case-sensitive

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

Note: Enclose values that contain delimiters (such as spaces) in double quotation marks.

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.

```
AGENT APPAGENT
URL service:jmx:rmi:///jndi/rmi://localhost:9999/server
MBEAN 'DefaultDomain:index=1,type=Config'
ATTRIBUTE cachesize
```

USER Statement—Specify a User ID

The USER statement specifies the user ID the agent job runs under.

Supported Job Types

This statement is required for the following job types:

- FTP (see page 96)
- HP Integrity NonStop (see page 105)
- Secure Copy (see page 174)
- Secure FTP (see page 175)

This statement is optional for the following job types:

- CPU Monitoring (see page 122)
- Database Monitor (see page 69)
- Database Stored Procedure (see page 71)
- Database Trigger (see page 76)
- Disk Monitoring (see page 123)
- File Trigger (see page 88)
- i5/OS (see page 113)
- IP Monitoring (see page 125)
- JMX-MBean Attribute Get (see page 53)
- JMX-MBean Attribute Set (see page 54)
- JMX-MBean Create Instance (see page 55)
- JMX-MBean Operation (see page 56)

- JMX-MBean Remove Instance (see page 57)
- JMX-MBean Subscribe (see page 58)
- PeopleSoft (see page 146)
- POJO (see page 60)
- Process Monitoring (see page 126)
- SQL (see page 81)
- Text File Reading and Monitoring (see page 127)
- UNIX (see page 190)
- Web Service (see page 211)
- Windows (see page 214)
- Windows Event Log Monitoring (see page 130)
- Windows Service Monitoring (see page 132)

Syntax

This statement has the following format for HP Integrity NonStop jobs:

```
USER group.user_ID
```

This statement has the following format for Windows and File Trigger jobs that monitor or access files on remote Windows systems:

```
USER domain\user_ID
```

This statement has the following format for all other job types that it applies to:

```
USER user_ID
```

group

Specifies the group that the user ID is in.

Limits: Maximum of 8 characters, case-sensitive; cannot contain delimiters (such as spaces)

domain

Specifies the domain that the user ID is in.

user_ID

Specifies the user ID to use when running the job.

Limits for HP Integrity NonStop jobs: Maximum of 8 characters, case-sensitive; cannot contain delimiters (such as spaces).

Limits for i5/OS jobs: Maximum of 10 characters, case-sensitive; cannot contain delimiters (such as spaces).

Limits for all other job types that USER applies to: Up to 128 characters; case-sensitive; cannot contain delimiters (such as spaces).

Notes:

- The USER statement overrides the default user ID defined on the scheduling manager.
- If you do not specify the USER statement and a default user ID is defined on the scheduling manager, the agent runs the job under the default user ID. If a default user ID is not defined on the scheduling manager, the agent runs the job under the default user ID defined on the agent. If a default user ID is not defined on the agent, the agent runs the job under the user ID that is running the agent.
- Each user ID requires a corresponding password. The password is encrypted and stored separately from the user ID. For more information on defining passwords and other security requirements, refer to your scheduling manager's documentation.

UNIX Jobs:

- The USER statement is required when running a binary file with the CMDNAME statement. The USER statement is optional with the SCRIPTNAME statement.
- If you use the agent security file to restrict the running of jobs to specific users, use the USER statement to define the user IDs you want the jobs to run under.
- If you run the agent as root and specify the USER statement, the agent can run the command or script with the user's environment and permissions. The environment is passed unchanged, as if the agent logged in as the specified user.
- If you run the agent as root, by default the parameter oscomponent.initialworking directory is set to the agent directory. To run a script or command, grant OTHER execution permission for the agent directory and the root home directory. Otherwise, you will get error message indicating that permission is denied.

Windows Jobs:

- To run Windows jobs under different user IDs, the agent must run as a Windows service under the local system account. For more information about configuring the agent to run as a Windows service, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.
- To run a job under a Windows user ID and password, you must define user ID and password pairs on the scheduling manager.

i5/OS Jobs:

- To use the USER statement, the agent must run under a profile that has *USE authority to the profile in the USER statement.
- If you specify the USER statement and no libraries have been specified with the LIBL statement, the agent defaults to using the default library list in the job's job description. However, you can specify that the job use the user's job description instead.

Database Trigger Jobs:

- The user that a Database Trigger job runs under must be authorized to create triggers on the database or schema the table belongs to.
- For Microsoft SQL Server, the user that a Database Trigger job runs under must own the database table identified by the TABLE_NAME statement.

Example: Specify a User ID in a Windows Job

This example runs sort.exe on the NT401 system under the user1 user ID:

```
AGENT NT401
USER user1
CMDNAME c:\payroll\sort.exe
```

Example: Specify a User ID on a Remote Windows Computer

This example runs calc.exe on the CYBNT server. CYBUSER is a user ID in the CYBDOM domain. CYBUSER is defined on the scheduling manager and has access permission to the public folder.

```
AGENT NT30
CMDNAME \\CYBNT\public\calc.exe
USER CYBDOM\CYBUSER
```

Example: Specify a User ID in a UNIX Job

This example runs the test1.ksh script on the UNIX_LA system under the user1 user ID:

```
AGENT UNIX_LA
SCIPTNAME /mfg/test1.ksh
USER user1
```

Example: Specify a User ID in an i5/OS Job

This example runs the MFGPROG program on the AS401 system under the user1 user ID:

```
AGENT AS401
CLPNAME MFGPROG
USER user1
```

Example: Specify a User ID in an HP Integrity NonStop Job

This example runs a command on an HP Integrity NonStop system under user ID "glsys", which is part of the "prod" group:

```
AGENT PROAGENT
COMMAND $C35.PROD.GETDEF
USER prod.glsys
```

USER_TYPE Statement—Specify the Oracle Database User Type

The USER_TYPE statement specifies the Oracle database user type. If you omit this statement from the job definition, the job runs with normal privileges.

Supported Job Types

This statement is optional for the following job types when an Oracle database is used:

- Database Monitor (see page 69)
- Database Stored Procedure (see page 71)
- Database Trigger (see page 76)
- SQL (see page 81)

Syntax

This statement has the following format:

```
USER_TYPE type
```

type

Specifies the type of Oracle user to log in as. The user must be defined in the Oracle database.

Limits: Up to 128 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- 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 statement.
- 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_TYPE statement overrides the default user type specified in the agent's agentparm.txt.
- USERTYPE is an alias of USER_TYPE.

Example: Specify a Database User Type

This example specifies a database user named dbuser1, who is logged in with sysdba privileges.

```
AGENT CYBDB1
USER dbuser1
USER_TYPE 'as sysdba'
SQL 'SELECT * from NEWORDS'
```

USERTYPE Statement—Specify the Oracle Database User Type

USERTYPE is an alias of USER_TYPE (see page 638).

USESETDEFAULTS Statement—Specify Whether Request Set Defaults Take Precedence over Program Defaults

The USESETDEFAULTS statement specifies whether request set defaults take precedence over concurrent program defaults in Oracle Applications.

Supported Job Type

This statement is optional for the Oracle E-Business Suite Request Set job type (see page 137).

Syntax

This statement has the following format:

```
USESETDEFAULTS Y|N
```

Y

Specifies that request set defaults take precedence over concurrent program defaults.

N

Specifies that concurrent program defaults take precedence over request set defaults. This is the default.

Notes:

- Your agent administrator can specify a default setting for all Oracle E-Business Suite Request Set jobs by setting the oa.default.useSetDefaultsFirst parameter in the agent's agentparm.txt file.
- The USESETDEFAULTS statement overrides the default setting specified in the oa.default.useSetDefaultsFirst parameter in the agent's agentparm.txt.

Example: Use Request Set Defaults

In this example, request set defaults take precedence over concurrent program defaults.

```
AGENT OAAGENT
REQUESTSET FNDRSSUB1310
OAUSER SYSADMIN
RESPNAME 'System Administrator'
APPLSHORTNAME BIS
USESETDEFAULTS Y
```

VARIANT Statement—Specify Variant Name

The VARIANT statement specifies the name of the variant. It is mandatory if the ABAP program requires it.

Supported Job Types

This statement is optional for the following job types:

- SAP Batch Input Session (see page 155)
- SAP R/3 (see page 166)

Syntax

This statement has the following format:

```
VARIANT variant
```

variant

Specifies the name of the variant. The value corresponds to the SAPGUI ABAP program Variant field on the Create Step dialog.

Limits: Up to 14 valid SAP characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- Use the VARIANT statement before the first ABAPNAME statement to set the default value for subsequent steps.
- Use the VARIANT statement within the boundaries of the step definition (between two ABAPNAME statements or between an ABAPNAME statement and the end of the job definition) to define the value for that step.

Example: Specify the Variant Name

This example specifies the variant named TEST for the BTCTEST ABAP:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
STARTMODE I
ABAPNAME BTCTEST
   STEPUSER user14
   VARIANT TEST
   SUCCESSMSG 'Program Selections'
```

WAITMODE Statement—Specify Whether to Monitor Conditions Immediately or Continuously

The WAITMODE statement specifies whether the job waits until the monitor conditions are met or tries to verify them immediately.

Supported Job Types

This statement is optional for the following job types:

- CPU Monitoring (see page 122)
- Disk Monitoring (see page 123)
- Text File Reading and Monitoring (see page 127)
- Windows Event Log Monitoring (see page 130)

Syntax

This statement has the following format:

```
WAITMODE NOW|WAIT
```

NOW

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails.

WAIT

Waits for the conditions to occur. When the conditions are met, the job completes.

Notes:

- When using the WAIT value in CPU Monitoring jobs, you must also specify the FROM operand, the TO operand, or both, in the CPU statement.
- When using the WAIT value in DISK Monitoring jobs, you must also specify the FROM operand, the TO operand, or both, in the DISK statement.
- When using the WAIT value in Text File Reading and Monitoring jobs, do not specify the TO operand in the SEARCHRANGE statement.

Example: Monitor CPU Usage Until It Falls in a Range

This example monitors CPU usage. The job completes when at least 20 percent and at most 90 percent of the CPU is being used.

```
AGENT MONAGT
CPU FROM(020) TO(090) USED
WAITMODE WAIT
```

Example: Monitor Disk Space Usage Until It Falls in a Range

This example monitors disk space usage. The job completes when at least 60 percent of disk space is available on the C drive.

```
AGENT MONAGT
DISK C FORMAT(PERCENT) FROM(60) TO(100) AVAILABLE
WAITMODE WAIT
```

WAKEHOST Statement—Specify the Computer to Ping

The WAKEHOST statement specifies the computer to ping after the Wake on LAN (WOL) signal is sent. You can specify the WAKEHOST statement, together with the WAKEPORTS and TIMEOUT statements, to confirm that the computer is awake.

Note: If you do not provide any ping information, the job completes immediately after the WOL signal is sent without verifying whether the signal worked.

Supported Job Type

This statement is optional for the Wake on LAN job type (see page 206).

Syntax

This statement has the following format:

```
WAKEHOST host
```

host

Specifies the host name or IP address of the computer to ping after the Wake on LAN (WOL) signal is sent.

Limits: Up to 100 characters; case-sensitive

Example: 172.24.36.107 (IPv4) or 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

Example: Broadcast the WOL Signal and Ping a Port

This example broadcasts the WOL signal to the server identified by the 172.16.00 IP address and the 00-1E-4F-C1-0F-FE MAC address. After the WOL signal is sent, the agent pings port 7 of the host computer to ensure it is available. If port 7 is available, the job completes successfully; otherwise, it fails. The job uses the default timeout for the ping.

```
AGENT AGENTNME
WAKEHOST host
WAKEPORTS 7
BROADCAST 172.16.0.0
MAC 00-1E-4F-C1-0F-FE
```

WAKEPASSWORD Statement—Specify the Wake on LAN Password

The WAKEPASSWORD statement specifies the Wake on LAN (WOL) password for network cards that support the Secure On security feature. After the agent broadcasts the WOL signal, the network card wakes up the computer only if the specified password is correct.

Note: The password is not encrypted on the scheduling manager.

Supported Job Type

This statement is optional for the Wake on LAN job type (see page 206).

Syntax

This statement has the following formats:

```
WAKEPASSWORD xx-xx-xx-xx
```

```
WAKEPASSWORD xx-xx-xx-xx-xx-xx
```

xx-xx-xx-xx

Specifies the Wake on LAN (WOL) password as a set of four 2-digit hexadecimal values separated by dashes.

Note: You can also separate the values using colons (:) or periods (.).

xx-xx-xx-xx-xx-xx

Specifies the Wake on LAN (WOL) password as a set of six 2-digit hexadecimal values separated by dashes.

Note: You can also separate the values using colons (:) or periods (.).

Example: Broadcast the WOL Signal Including a Password

This example broadcasts the WOL signal including a password to the server identified by the 172.16.00 IP address and the 11-22-33-44-55-66 MAC address. If the specified password matches the password stored on the server's network card, the server wakes up.

```
AGENT AGENTNME
BROADCAST 172.16.0.0
MAC 11-22-33-44-55-66
WAKEPASSWORD AA-BB-CC-DD-EE-FF
```

WAKEPORTS Statement—Specify the Ports to Ping

The WAKEPORTS statement specifies the ports to ping after the Wake on LAN (WOL) signal is sent. You can specify the WAKEPORTS statement, together with the WAKEHOST and TIMEOUT statements, to confirm that the computer is awake. If at least one of the specified ports is available, the job completes successfully; otherwise, it fails.

Note: If you specify the WAKEHOST statement and do not specify the WAKEPORTS statement, the following ports are pinged by default: 21, 22, 23, 80, 111, 135, 139, and 445.

Supported Job Type

This statement is optional for the Wake on LAN job type (see page 206).

Syntax

This statement has the following format:

```
WAKEPORTS port|'port,port,...'
```

port

Specifies a port to ping after the agent sends the Wake on LAN (WOL) signal.

Notes:

- To specify multiple ports, separate each port with a comma and enclose the entire list of ports in single quotes.
- The entire value can be up to 1024 characters.

Example: Broadcast the WOL Signal and Ping a Port

This example broadcasts the WOL signal to the server identified by the 172.16.00 IP address and the 00-1E-4F-C1-0F-FE MAC address. After the WOL signal is sent, the agent pings port 7 of the host computer to ensure it is available. If port 7 is available, the job completes successfully; otherwise, it fails. The job uses the default timeout for the ping.

```
AGENT AGENTNME
WAKEHOST host
WAKEPORTS 7
BROADCAST 172.16.0.0
MAC 00-1E-4F-C1-0F-FE
```

WEBPOSTING Statement—Specify Whether to Post the SAP Job Log and Spool on the Web

The WEBPOSTING statement specifies whether to post the SAP job log and spool on the Web.

Supported Job Type

This statement is optional for the following job types:

- SAP Job Copy (see page 164)
- SAP R/3 (see page 166).

Syntax

This statement has the following format:

```
WEBPOSTING Y|N
```

Y

Posts the SAP job log and spool on the web.

N

Does not post the SAP job log and spool on the web.

Example: Post the SAP Job Log and Spool on the Web

This example posts the SAP job log and spool on the Web:

```
SAPJOBNAME SAPTEST
AGENT SAPHTAGENT
WEBPOSTING Y
ABAPNAME BTCTEST
```

WSDL_URL Statement—Specify a WSDL URL

The WSDL_URL statement 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 statement is optional for the Web Service job type (see page 211).

Syntax

This statement 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 1024 characters; case-sensitive

Notes:

- Enclose values that contain delimiters (such as spaces) in double quotation marks.
- In a Web Service job, if you specify the WSDL_URL statement but not the ENDPOINT_URL statement, you must specify both the SERVICENAME and PORTNAME statements.

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.

```
AGENT WSAGENT
TARGETNAMESPACE http://www.webserviceX.NET/
SERVICENAME StockQuote
PORTNAME StockQuoteSoap
OPERATION GetQuote
WSDL_URL http://www.webservicex.com/stockquote.asmx?WSDL
ENDPOINT_URL http://www.webservicex.com/stockquote.asmx
PARAMETER TYPE(xsd:string) VALUE(CA)
RETURNCLASS java.lang.String
RETURNXML string
RETURNNAMESPACE http://www.webserviceX.NET/
```