CA Workload Automation DE

Define Perspective Help

r11.3 SP2

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

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

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

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

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

The manufacturer of this Documentation is CA.

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

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Process Automation
- CA Spectrum® Service Assurance (CA Spectrum SA)
- CA Workload Automation AE
- CA Workload Automation Agent for Application Services (CA WA Agent for Application Services)
- CA Workload Automation Agent for Databases (CA WA Agent for Databases)
- CA Workload Automation Agent for i5/OS (CA WA Agent for i5/OS)
- CA Workload Automation Agent for Informatica (CA WA Agent for Informatica)
- CA Workload Automation Agent for Linux (CA WA Agent for Linux)
- CA Workload Automation Agent for Micro Focus (CA WA Agent for Micro Focus)
- CA Workload Automation Agent for Microsoft SQL Server (CA WA Agent for Microsoft SQL Server)
- CA Workload Automation Agent for Oracle E-Business Suite (CA WA Agent for Oracle E-Business Suite)
- CA Workload Automation Agent for PeopleSoft (CA WA Agent for PeopleSoft)
- CA Workload Automation Agent for Remote Execution (CA WA Agent for Remote Execution)
- CA Workload Automation Agent for SAP (CA WA Agent for SAP)
- CA Workload Automation Agent for UNIX (CA WA Agent for UNIX)
- CA Workload Automation Agent for Web Services (CA WA Agent for Web Services)
- CA Workload Automation Agent for Windows (CA WA Agent for Windows)
- CA Workload Automation CA 7 Edition
- CA Workload Automation DE
- CA Workload Automation DE Web Client
- CA Workload Automation Desktop Client (CA WA Desktop Client)
- CA Workload Automation ESP Edition
- CA Workload Automation High Availability DE (CA WA High Availability)
- CA Workload Automation Restart Option EE (CA WA Restart Option)
- CA Workload Automation Web Services (CA WA Web Services)
- CA Workload Control Center (CA WCC)

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.

Contents

Chapter 1: Working with Applications 13

Applications 13
- How the Server Processes the Application 15
- Application Versions 16
- Concurrent Application Generations 18
- SubApplications 18
- Anticipated End Times 19
- Critical Path 19
- Application Templates 21
- Application Integration Tags 25

How to Define an Application 26
- Specify the Application Properties 28
- Schedule the Application Using an Event 29
- Add the Jobs to the Application Workspace 30
- Draw the Dependencies Between the Jobs 31
- Define Job Details of Each Job 32
- Upload the Application 33
- Simulate the Event 34

Locate a Job in the Application Workspace 36
Edit the Job Definitions of Multiple Jobs in an Application 37
Download an Application 40
Delete an Application 41
Lock an Application 42
Unlock an Application 43
Set Application Defaults 44
Add Comments to an Application 45
Delay Application Generations from Running Until the Current Generation Completes 45
Prevent Events from Triggering While the Current Generation is Running 46
Delay Each Job in an Application from Running Until the Job Completes in a Previous Application Generation 47
Enable Anticipated End Times and Critical Path Analysis 48
Suppress Event Trigger Notifications When No Work is Selected 49
Set the Reason Option for Job Commands as Mandatory 50
Export Application Parameters 51
Map an Application to a Business Service 52
Set the Default Zoom Settings for Application Graphs 54

Chapter 2: Working with Workload Objects 55

Workload Objects 56
- Agents 56
- Agent Groups 58
- Run Frequency 59
- On-Request Jobs 60
- Conditional Jobs 61
- Job Profiling 62
- Using a Namespace for a User that has Different Passwords 63
- Continuous Monitoring Usage 64
- Job Output Variables 65
- Payload Producing and Payload Consuming Jobs 71

Locate a Job on a Server 73
Specify Run Frequency in a Workload Object 75
Run a Job on an Agent Group for Load Balancing 77
Run a Job on All Agents Defined in a Group 78
Pass Arguments to a Script or Command 79
Define Job Success or Failure 80
Specify Environment Variables 82
Add Comments to a Job 85
Set Job Defaults 85
Resubmit a Failed Job Automatically 87
Define the Jobs in a SubApplication 88
Define an On-Request Job 89
Define a Conditional Job 89
Delay a Job from Running Until the Job Completes in a Previous Application Generation 90
Override a Job's Average Execution Time 91
Identify a Critical Job in an Application 92
Specify a Profile for a Job 93
Application and Web Services Jobs 95
- Entity Bean Jobs 97
- HTTP Jobs 111
- JMS Publish and JMS Subscribe Jobs 117
- JMX Jobs 128
- POJO Jobs 154
- RMI Jobs 161
- Session Bean Jobs 165
- Web Service Jobs 175

CA WA Jobs 200
- Create a Link 201
- Create a Task 202

Database Jobs....202
Define a DB Stored Procedure Job....203
DB Monitor and DB Trigger Jobs....209
Define an SQL Job....216
External Jobs....219
Define an External Job....219
File Transfer Jobs....223
FTP Jobs....223
Define an SCP Job....239
Define an SFTP Job....243
Informatica Jobs....247
Define an Informatica Job....247
Micro Focus Jobs....255
Define a Micro Focus Job....256
Monitoring Jobs....263
Define a CPU Monitoring Job....264
Define a Disk Monitoring Job....269
File Trigger Jobs....274
Define an IP Monitoring Job....288
Define a Process Monitoring Job....290
Define a Text File Reading and Monitoring Job....293
Define a Windows Event Log Monitoring Job....301
Define a Windows Service Monitoring Job....306
MSSQL Server Jobs....308
Define a MSSQL Server Job....308
Oracle E-Business Suite Jobs....320
Define a Copy Single Request Job....321
Define a Request Set Job....324
Define a Single Request Job....337
Search for an Oracle Applications Application....349
PeopleSoft Jobs....350
Define a PeopleSoft Job to Run a Request....352
Distribute PeopleSoft Reports....357
Remote Execution Jobs....358
Define a Remote Execution Job....359
SAP Jobs....362
Set Up a Connection to Your SAP System....363
Filtering and Listing Jobs on Your SAP System....364
Define an SAP-Batch Input Session Job....375
Define an SAP-BW Info Package Job....380
Define an SAP-BW Process Chain Job....382
Define an SAP-Data Archiving Job....384

Define an SAP-Event Job 387
Define an SAP-Job Copy Job 389
Define an SAP-Process Monitor Job 393
Define an SAP-R3 Job 396
SNMP Jobs 406
Define an SNMP Subscribe Job 407
Define an SNMP Trap Send Job 409
Define an SNMP Value Get Job 418
Define an SNMP Value Set Job 425
System Jobs 434
i5/OS Jobs 435
Define an OpenVMS Job 446
Define a Tandem NSK Job 448
UNIX Jobs 451
Wake on LAN Jobs 462
Windows Jobs 467
z/OS Jobs 477
z/OS-Data Set Trigger Jobs 478
Define a z/OS-Manual Job 486
Define a z/OS-Regular Job 488

Chapter 3: Working with Events 493

Events 493
Event Triggers 494
Event Schedule Criteria 495
Expect Times 496
Workload Simulation 496
Suspend and Resume Times 497
Event Priority 498
Date-Time/Manual Events 499
File Trigger Events 500
z/OS Data Set Trigger Events 501
JMS Subscribe Events 502
SAP Event Monitor Events 503
Database Monitor and Database Trigger Events 504
Variable Dependency Monitor Events 505
Create an Event 505
Locate a Job in the Graphical Simulation 508
Specify Suspend and Resume Times 509
Prevent an Event from Triggering If an Application is Active 511
Schedule Workload 512

Specify Application Parameter Values in a Date-Time/Manual Event 514
Remove Application Parameters from a Date-Time/Manual Event 515
Trigger Workload When a File is Created 516
Trigger Workload Based on z/OS Data Set Activity 519
Trigger Workload When a JMS Message is Received 526
Trigger Workload When an SAP Event is Raised 529
Trigger Workload When the Number of Rows in a Database Table Changes 532
Trigger Workload When a Row is Added, Deleted, or Updated in a Database Table 535
Trigger Workload When a Global Variable Expression is Met 542

Chapter 4: Managing Predecessor Dependencies 547

Predecessor Dependencies 547
Interlinked Dependencies 550
Manual Dependencies 551
Cross-Application Dependencies 552
Inherited Dependencies 553
Edit a Job's Predecessor and Successor Dependencies Using a Dialog 555
Edit the Release Condition of a Predecessor Dependency 556
Edit the Release Conditions of a Job's Predecessor and Successor Dependencies 558
Override Job Inheritance 559

Chapter 5: Managing Time Dependencies 561

Time Dependencies 562
Job Submission Time 563
Overdue Jobs 563
Premend Jobs 568
Delay Job Submission 569
Mark a Job Overdue if It Starts Late 571
Mark a Job Overdue if It Ends Late 572
Mark a Job Overdue if It Runs Long 573
Mark a Job Premend if It Ends Prematurely 574
Abandon Predecessor Dependencies at a Specified Time 575
Abandon Resource Dependencies at a Specified Time 576
Abandon Variable Dependencies at a Specified Time 577
Bypass Jobs at a Specified Time 578
Propagate Dueout Times 579

Chapter 6: Managing Resource Dependencies 581

How Jobs with Resource Dependencies are Submitted 581
Job Priority 582

Job Reservation 582
Job Priority and Reservation 583
Set Up Resource Dependencies 584
Set a Job's Priority 586
Reserve a Resource 586
Example: Prevent Jobs from Running Concurrently 587
Example: Control Concurrent Access to a Database 588
Example: Control When Low-Priority Jobs Run 589

Chapter 7: Managing Global Variable Dependencies 591

Global Variable Dependencies 591
How Jobs with Global Variable Dependencies are Submitted 592
Set Up a Variable Dependency 592
Modify a Variable Dependency 597
Delete a Variable Dependency 600
%VAR Statement—Specify Global Variables in Job Fields 600
Example: Specify a Global Variable in a Database Job Definition 602

Chapter 8: Working with Notifications 605

Notifications 605
Monitor States 606
Email Notifications 608
Alert Notifications 609
SNMP Notifications 609
Set Up an Email Notification in the Job Definition 610
Set Up an Email Notification in the Application 613
How to Set Up an Alert Notification 616
Define an Alert that Triggers the Event 617
Define an Alert that Runs the JavaScript Script 618
Set Up an Alert Notification in the Job Definition 620
Set Up an Alert Notification in the Application 623
Create an Event that the Alert Triggers 624
Example: Resubmit a Job if it Completes with a Particular Exit Code Before a Specific Time 626
Set Up an SNMP Notification in the Job Definition 628
Set Up an SNMP Notification in the Application 630

Chapter 9: Working with JavaScripts 633

JavaScripts 633
Store a JavaScript Script in the Application 634
Specify JavaScript Scripts in the Event Definition 636

Specify a JavaScript Script in the Application Definition 637
Specify a JavaScript Script in the Job Definition 638
%IF Statement—Create Conditional Schedule Criteria 640

Chapter 10: Working with Schedule Criteria 643

Schedule Criteria 644
Days of the Week and Months 645
Dates 646
- Ordinal Numbers 647
- Days of Months 647
- Range of Days 647
- Implied Periods 648
- Date Format 648

Times 650
Time Zones 651
Daylight Saving Time 653
- Spring-Forward Adjustments 654
- Fall-Backward Adjustments 656

Appendix A: Using Scheduling Terms 657

Scheduling Terms 657

Index 663

Chapter 1: Working with Applications

This section contains the following topics:

Applications (see page 13)
How to Define an Application (see page 26)
Locate a Job in the Application Workspace (see page 36)
Edit the Job Definitions of Multiple Jobs in an Application (see page 37)
Download an Application (see page 40)
Delete an Application (see page 41)
Lock an Application (see page 42)
Unlock an Application (see page 43)
Set Application Defaults (see page 44)
Add Comments to an Application (see page 45)
Delay Application Generations from Running Until the Current Generation Completes (see page 45)
Prevent Events from Triggering While the Current Generation is Running (see page 46)
Delay Each Job in an Application from Running Until the Job Completes in a Previous Application Generation (see page 47)
Enable Anticipated End Times and Critical Path Analysis (see page 48)
Suppress Event Trigger Notifications When No Work is Selected (see page 49)
Set the Reason Option for Job Commands as Mandatory (see page 50)
Export Application Parameters (see page 51)
Map an Application to a Business Service (see page 52)
Set the Default Zoom Settings for Application Graphs (see page 54)

Applications

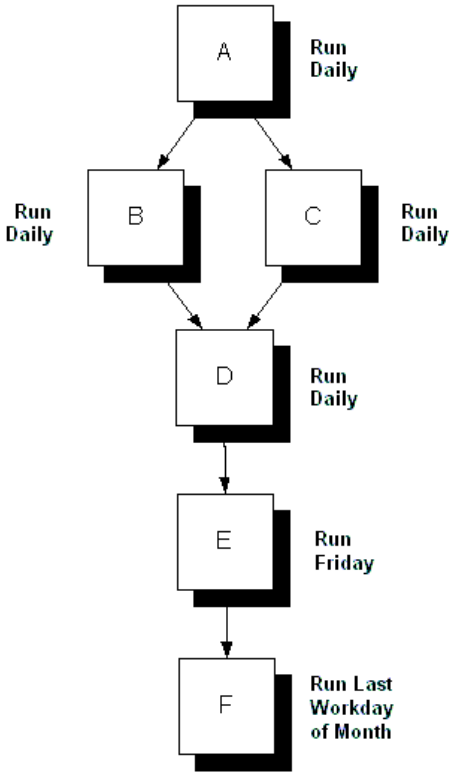
An Application consists of one or more jobs and other workload objects. Applications let you group related jobs and define their dependencies. For example, you may want to group all your payroll jobs in one Application. Jobs in an Application may run on different platforms.

By default, the server submits a job after all of the job's predecessors complete successfully. Predecessors are workload objects that must complete before another workload object can run. Successors are workload objects that must wait for the completion of predecessors before they can run.

Example: Predecessors and Successors in an Application

Every day, jobs A, B, C, and D run. On Fridays, job E also runs. On the last workday of the month, job F also runs.

The following illustration shows the days on which the various jobs run:



The following table lists the predecessors and successors of each job.

Job	Predecessors	Successors
A	None	B, C
B	A	D
C	A	D
D	B, C	E
E	D	F
F	E	None

How the Server Processes the Application

An Application passes through the following two phases:

Generation phase

Generates an instance of an Application triggered by an Event. This instance is named a generation. During this phase, the server does the following:

- Selects the jobs that will run in the Application based on their run frequencies and the calendars specified in the Event
- Builds the Application with the selected jobs

Process phase

Processes the jobs after selecting the jobs to run in the Application. During this phase, the server does the following:

- Checks for any predecessor dependencies at the job submission time

  Note: Each job has a hold count corresponding to the number of immediate predecessors. A job cannot be submitted if its hold count exceeds zero.
- Subtracts one from a job's current hold count after one of the job's predecessors complete successfully
- Checks the job for any resource dependencies after a job's hold count reduces to zero
- Submits the job after all required resources are available
- Completes the Application after all jobs in the Application are complete or are bypassed

Example: Building an Application on Different Days

Every day, jobs A, B, C, and D run in an Application. On Fridays, job E also runs. On the last workday of the month, job F also runs.

In this Application, one of the following will occur:

- If today is not a Friday and not the last workday of the month:
  - Job B and job C run after job A completes successfully
  - Job D runs after job B and job C complete successfully
  - Jobs E and F do not run
- If today is a Friday but not the last workday of the month
  - Job E runs after jobs A, B, C, and D complete successfully
  - Job F does not run
- If today is the last workday of the month but not a Friday
  - Job F runs after jobs A, B, C, and D complete successfully
  - Job E does not run
- If today is a Friday and the last workday of the month
  - Job E runs after jobs A, B, C, and D complete successfully
  - Job F runs after job E completes successfully

Application Versions

The server maintains a list of versions for each Application. When you schedule an Application, the server uses the latest version uploaded to the server. With Application versioning, you can list, open, modify, delete, and run old Application versions.

Note: There is no limit to the number of Application versions kept.

More information:

Download an Application (see page 40)
Delete an Application (see page 41)

Application Locking and Unlocking

The server manages concurrent updates to an Application. Sometimes collisions can occur if two users try to upload the same Application around the same time.

Example: Management of Concurrent Updates to an Application

- On machine X, user SCHEDMASTER downloads Application APPL1.
- On machine Y, user GURU downloads Application APPL1 also.
- SCHEDMASTER updates and uploads APPL1.
- GURU updates and tries to upload APPL1. The upload fails because SCHEDMASTER uploaded the Application after GURU downloaded it.

When collisions such as the above occur, the server provides the following recovery options:

- GURU can force an upload without considering the changes made by SCHEDMASTER. With this option, APPL1 contains only the changes made by GURU.
- GURU can download the Application version uploaded by SCHEDMASTER, update the Application, and retry to upload. With this option, APPL1 contains the changes made by both GURU and SCHEDMASTER.
- GURU can abandon the upload. With this option, APPL1 contains only the changes made by SCHEDMASTER.

When you lock an Application, other users (with different user IDs) cannot automatically update and upload the Application until you manually unlock the Application. If a collision occurs, the same recovery options as in the example apply.

More information:

Lock an Application (see page 42)
Unlock an Application (see page 43)

Concurrent Application Generations

By default, if two generations of an Application are active on the system at the same time, they run concurrently. In this case, the second generation may complete before the first generation completes and new generations may run before their previous generations complete.

You can prevent two generations of the same Application from running concurrently by doing one of the following:

- Delay a new generation of the Application from running until the current generation completes
- Prevent all Events associated with the Application from triggering while the current generation is running (Application-level)
- Prevent a specific Event associated with the Application from triggering while the current generation is running (Event-level)

You can also delay a job from running until the job completes in a previous Application generation.

SubApplications

An Application can contain multiple subApplications. SubApplications are groups of workload objects that belong to a single Application. You can use subApplications to break up a large Application into smaller, easier to manage groups of jobs.

You can issue a command to perform the following actions against all jobs in a subApplication:

- Bypass or unbypass all the jobs in a subApplication
- Request or unrequest all on-request jobs in a subApplication
- Hold or release all the jobs in a subApplication
- Complete all the jobs in a subApplication
- Remove all the jobs in a subApplication from wait status

Example: Bypass a SubApplication

An Application contains several jobs that run on the weekend. On some weekends, these jobs should not run. By putting these jobs in a subApplication, you can bypass them with a single command.

More information:

Define the Jobs in a SubApplication (see page 88)

Anticipated End Times

If the Estimate end time or Propagate dueout time option is enabled in the Application properties, the server calculates the anticipated end times for all jobs in the Application when it generates the Application. The server recalculates anticipated end times every time the state of a job changes in the Application. You can review the Application, at any stage, and determine when any job is expected to complete.

The server calculates the anticipated end times of jobs based on Event submission time, predecessor dependencies, time dependencies, and the job's average execution time. You can override the average execution time the server uses to calculate a job's anticipated end time when you define the job or modify the job definition.

Note: If a job does not have an average execution time when it runs, the server does not calculate the anticipated end time of the job. In future runs, the server calculates the anticipated end time of the job based on the historical execution times of the job.

More information:

Enable Anticipated End Times and Critical Path Analysis (see page 48)
Override a Job's Average Execution Time (see page 91)
Overdue Jobs (see page 563)

Critical Path

By default, the server calculates an Application's critical path based on the job that finishes last in the Application. You can also identify a job in an Application as critical. The longest path to that job, based on historical execution time, is the critical path.

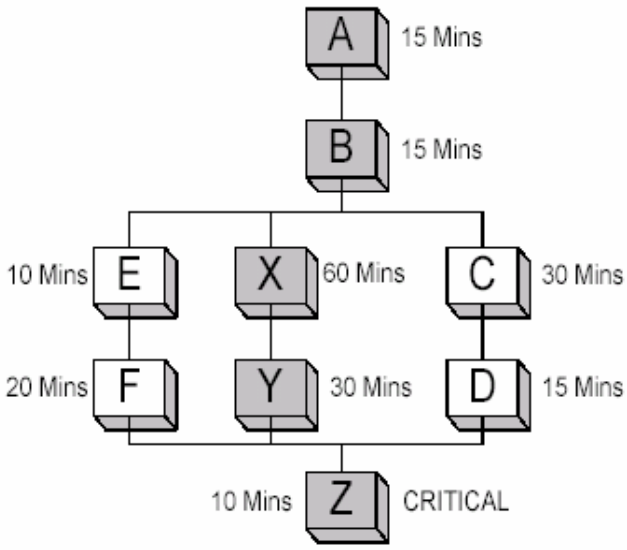
The server determines the critical path when it generates an Application. The critical path is dynamic; it changes as the Application runs.

Note: If none of the jobs in the Application have an average execution time when the Application runs, the server does not calculate the critical path of the Application. In future runs, the server calculates the critical path based on the historical execution times of each job in the Application.

Example: Critical Path of an Application

An Application has historical execution times for each job. The longest path to job Z is 130 minutes through jobs A, B, X, Y, and Z. The critical path in this Application, at this time, consists of jobs A, B, X, Y, and Z.

The following illustration displays the Application's critical path:



The critical path for an Application can vary depending on the jobs selected.

The following table displays the Application's critical path when certain jobs are not selected:

Jobs Not Selected	Critical Path
X, Y	A, B, C, D, Z
X, Z	A, B, C, D
X, Y, D	A, B, E, F, Z, and A, B, C, Z
Z	A, B, X, Y

More information:

Enable Anticipated End Times and Critical Path Analysis (see page 48)
Identify a Critical Job in an Application (see page 92)

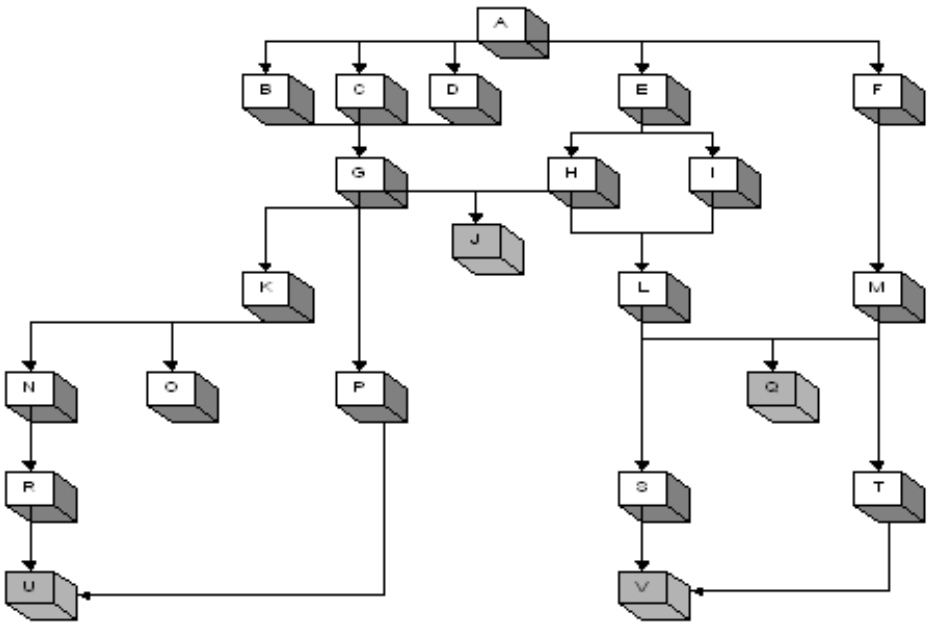
Multiple Critical Paths

An Application may have more than one critical path. You can also identify one or more jobs as critical in an Application.

Example: Critical Jobs in an Application

Four jobs J, Q, U, and V (represented by grayed boxes) are critical jobs. The server identifies the longest paths to these jobs as critical paths, based on historical execution times.

The following illustration displays the critical jobs in the Application:



Application Templates

You can create Application templates for third-party users of CA Workload Automation DE to simplify the process of running their workload. Application templates are ready-to-use Application definitions that third-party users can run at anytime by passing required parameters. For example, you can define a single Application template for the payroll processes of two companies so that each company runs the Application template with their own parameters.

Application templates provide increased abstraction for third-party users in running their workload. They can run workload without being familiar with CA Workload Automation DE Events or JavaScript. Users can focus more on their business applications rather than the underlying CA Workload Automation DE technology.

Specifying Application Parameters

You must define Application parameters as symbolic variables in the EVAR context. The syntax for defining Application parameters is as follows:

%EVAR.*parameter_name*

%EVAR

Specifies a symbolic variable in the EVAR context.

parameter_name

Specifies the Application parameter name.

Notes:

- Application parameter names can contain alphanumeric characters and underscores, but the first character cannot be numeric.
- Application parameters can be concatenated with strings or other Application parameters, for example, %(EVAR.emp_name)PayCalc or %EVAR.emp_ID.%EVAR.emp_NAME.
- Application parameter names are case-sensitive. For example, employee_name and Employee_name are two different parameters.

Example: Specifying an Agent Name as an Application Parameter

To specify an Agent name as an Application parameter

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click a job, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Ensure that the Agent section is expanded.
4. Enter **%EVAR.application_agent** in the Agent field.
5. Click OK.

   The Agent name is defined as an Application parameter.

More information:

Export Application Parameters (see page 51)

How Application Templates are Defined and Used

The following sequence explains how you create Application templates and third-party users use them.

1. You define an Application template on CA WA Desktop Client that includes the parameters to pass when the Application is executed. For example, you can define a payroll processing Application that accepts two parameters named employee_count and date_of_run.
2. You define the lifecycle of the Application using the following web service functions:

   runApplicationCreate

   Creates an Event to run an Application. The runApplicationCreate function specifies the Application to run, when the Application runs, and the argument values to be passed to the Application.

   runApplicationRead

   Reads an Event created by the runApplicationCreate function.

   runApplicationUpdate

   Updates an Event created by the runApplicationCreate function.

   runApplicationDelete

   Deletes an Event created by the runApplicationCreate function.

   runApplicationList

   Lists Events created by the runApplicationCreate function.

   Note: For more information about web service functions, see the *Programming Guide*.
3. You make the Application template available to third-party users.
4. Third-party users start the Application template either as a web service front end or a back-end server process by passing the required parameters.
5. The Application template creates an Application instance and a token for each third-party user by calling the runApplicationCreate web service function. Once created, the Application instances run with the same parameter values each time. For example, if a third-party user runs the payroll processing Application template with 100 employees on the first day of a month, the corresponding Application instance runs on the *same* day of each month for the *same* number of employees.

6. If a third-party user wants to modify the parameter values, they can use the corresponding Application instance token and update the Application instance. The Application template uses the runApplicationRead web service function to read the existing Application instance. When the user reruns the Application instance with the modified values, the Application template calls the runApplicationUpdate web service function to create a modified Application instance. For example, a third-party user might run the payroll processing application for a reduced number of employees because of a workforce reduction. The modified Application instance runs for the updated employee count for the subsequent months.
7. If a third-party user no longer needs the Application template, they can use the runApplicationDelete web service function to delete the Application instance.

Example: Using an Application Template for Inventory Management

Suppose that you want to run an inventory check of your accessory business on December 31 at 9:00 a.m. You can define a simple Application with two parameters, "category=Perfumes" and "brand=Premium".

You do not need to define any calendars for the Application. You can define the two parameters as %EVAR.category and %EVAR.brand and invoke the runApplicationCreate web service function with the following arguments:

- applicationName—INVENTORY
- scheduleStatements—DEC 31 YYYY 09:00
- noscheduleStatements—NULL
- Calendars—NULL
- arguments—category=Perfumes, brand=Premium

The web service function returns a tag, say xyz, that represents the workload to be executed. The INVENTORY Application is generated and starts execution on December 31, YYYY at 09:00 a.m.

You can delete the Application by calling the runApplicationDelete web service function with eventName=xyz.

Application Integration Tags

Application integration tags let you expose Application-level information to other CA products. For example, you can expose Application state data that can be managed by a consuming CA product such as CA Spectrum SA.

To integrate CA Workload Automation DE with CA Spectrum SA, the CA Workload Automation DE connector must be installed. The connector retrieves Application state data from the CA Workload Automation DE server, transforms the data into a common format, and exposes the data for CA Spectrum SA to manage. CA Spectrum SA consolidates data retrieved by the connector and lets you model services to represent cross-product data based on logical business functions.

Note: For more information about the CA Workload Automation DE connector, see the *CA Workload Automation DE Connector Guide* packaged with the connector.

Example: Using Application Tags to Map an Application to Business Services

Suppose that an Application named Backup is defined on the CA Workload Automation DE server. A company invokes Backup every week to back up all its critical database servers. A separate instance of the Backup Applications is triggered for each critical database.

If a problem occurs during the backup of a critical database, DatabaseServices and ITServices groups are impacted. In the company's CA Spectrum SA system, these groups are represented as two distinct business services named DBServices and ITServices.

To associate the Backup Application with each business service, you specify the following Application integration tags in the Application definition:

- ssa:DBServices
- ssa:ITServices

For example, assume that five instances of the Backup Application are running at the same time. If a problem occurs during the Application execution of one of those instances, CA Spectrum SA generates Major alerts against the DBServices and ITServices business services to indicate service degradation. Using the information provided in the alert, users of CA Spectrum SA can determine which instance of the Backup Application requires attention and take the appropriate action. When the problem is resolved, CA Spectrum SA generates Normal alerts, overriding the previously issued Major alerts.

More information:

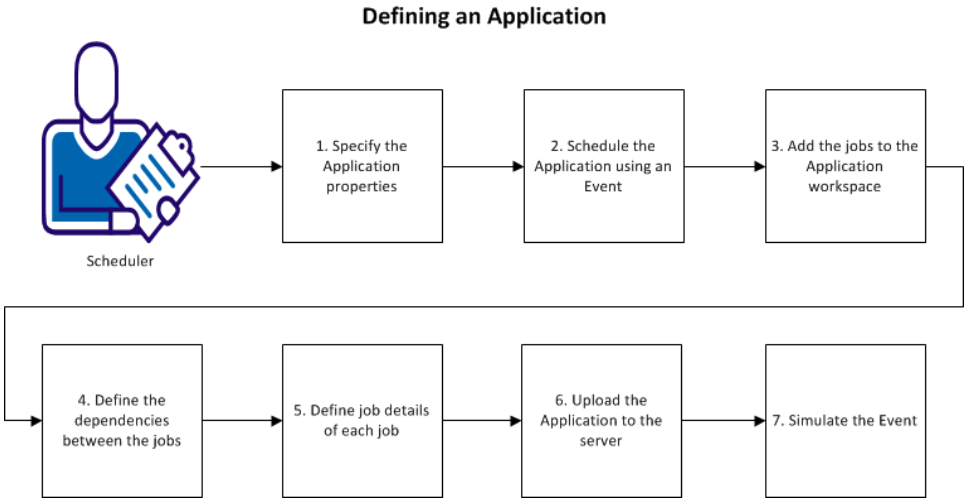
Map an Application to a Business Service (see page 52)

How to Define an Application

As a scheduler, you are responsible for defining the jobs in the Application and their dependencies. An Application consists of one or more jobs and other workload objects. Applications let you group related jobs and define their dependencies. For example, you can group all your payroll jobs in one Application. Jobs in an Application can run on different platforms.

By default, the server submits a job after all the predecessors of the job complete successfully. Predecessors are workload objects that must complete before another workload object can run. Successors are workload objects that must wait for predecessors to complete before they can run.

This scenario walks you through the process of defining an Application:

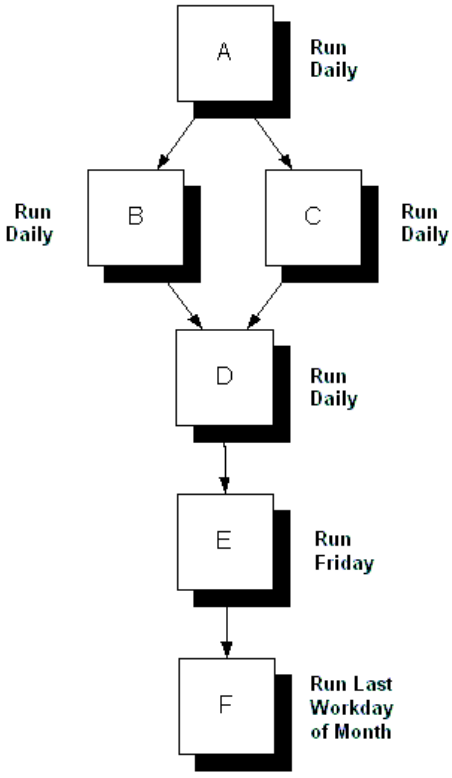


1. Specify the Application properties (see page 28).
2. Schedule the Application using an Event (see page 29).
3. Add the jobs to the Application workspace (see page 30).
4. Define the dependencies between the jobs (see page 31).
5. Define job details of each job (see page 32).
6. Upload the Application to the server (see page 33).
7. Simulate the Event (see page 34).

Example: Defining an Application

Every day, jobs A, B, C, and D run. On Fridays, job E also runs. On the last workday of the month, job F also runs.

The following illustration shows the days on which the various jobs run:



Specify the Application Properties

You can specify general Application properties and set defaults for the jobs. For example, you can prevent Application generations from running concurrently or can specify a default run frequency and the agent name. When defining the job details of each job, you can use the default run frequency and the agent name. Or you can override the defaults that are specified in the Application properties.

Note: The defaults that you define in the Application properties apply to all jobs in the Application, regardless of the job type. If you want to specify a default for a particular job type, use job defaults.

Follow these steps:

1. Connect to the server as a scheduler using CA WA Desktop Client.
2. Open the Define perspective.
3. Right-click the server connection in the Application Workspace view, and select New from the pop-up menu.

   The Basic page of the Application properties dialog opens.
4. Specify an Application name in the Name field.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), dollar sign ($), and underscore (_).
5. Complete the rest of the fields as required.

   Note: By default, the server uses the Application name for monitoring the Application. To override the default, specify a name in the Runtime name field. You can use a symbolic variable in the Runtime field.
6. Click OK.

   The Application appears in the workspace.

Schedule the Application Using an Event

When you define a new Application, CA WA Desktop Client defines a default Date-Time/Manual Event. You can use this Event to schedule the Application or to run the Application manually. You can replace the default Event with another Event or can define more Events as required.

Follow these steps:

1. Right-click the Date-Time/Manual Event in the Application Events view, and select Edit from the pop-up menu.

   The Properties page of the Event definition dialog opens.

2. Complete the fields as required.

   Note: The Event prefix and name uniquely identify an Event on the system where it resides. If you do not specify a calendar, the server uses the SYSTEM calendar**.**

3. Click Schedule in the left pane.

   The Schedule page opens in the right pane.

4. Click Add Schedule.

   A new row is added in the Specify schedule criteria table.

5. Click the ellipses (…) in the When field to build and test your scheduling criteria.

   Note: If you are familiar with the scheduling terms, type your scheduling criteria in the When field, for example, 3AM, HOURLY, or 10AM DAILY.

   The Schedule event dialog opens.

6. Select the Use generated statement option button and choose scheduling terms from the lists provided. Or select the Use free format statement option button and type your scheduling criteria in the text box.

7. Click Test to test your scheduling criteria.

   The Test results text box displays the date and time of the next ten Event executions.

8. Click OK to accept your scheduling criteria.

   The When field displays your scheduling criteria.

9. Click OK.

10. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

    If the Event contains no errors, it is saved. The server schedules your Event.

More information:

Date-Time/Manual Events (see page 499)
Event Schedule Criteria (see page 495)

Add the Jobs to the Application Workspace

You can add as many jobs to your Application as required.

Follow these steps:

1. Select the workload object that represents the job you want to add from the Palette view.

   You may have to open a job group (drawer) to select the job. For example, to select the UNIX job, select the System group from the Palette view and select the UNIX job in the System group.

   Note: You can limit which workload objects appear on the Palette view. For example, if you do not schedule SAP jobs, right-click the Palette view, select Customize from the pop-up menu, select the SAP group in the Customize Palette dialog, select the Hide check box, and click OK.

2. Drag the workload object onto the workspace.

   The job icon appears on the Application workspace view.

3. Drag all the required jobs from the Palette view to the Application workspace view. The following positioning rules apply:

   - If a job must run after another job, place it below that job on the workspace.
   - If a job must run before another job, place it above the job on the workspace.
   - If a job must run at the same time as another job, place it beside that job on the workspace.

   The jobs are added to the Application.

Draw the Dependencies Between the Jobs

You can create predecessor dependencies between the jobs that are defined for your Application by drawing the dependencies in the Application workspace.

Note: In an Application that contains many jobs, you might find it easier to create the dependencies between the jobs by selecting a job's predecessor and successor dependencies through a dialog instead. To use this feature, right-click the job in the workspace, and select Edit Job Dependencies from the pop-up menu.

Follow these steps:

1. Click Dependencies at the bottom of the workspace.
2. Select a predecessor and drag the mouse to the next job to be released (successor).

   A dependency line appears between the two jobs.
3. Repeat to draw all the dependencies.

   The dependencies between all the jobs in the Application are created.
4. Click Layout at the bottom of the workspace to organize the layout of your jobs.

   The layout of your jobs is organized.

More information:

Edit a Job's Predecessor and Successor Dependencies Using a Dialog (see page 555)

Define Job Details of Each Job

You must define all required details for all the jobs of your Application. You can also override defaults and specify optional details such as time dependencies, notifications, and resource requirements.

Follow these steps:

1. Right-click a job, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

2. Ensure that the Agent section is expanded.

   Note: Some sections are collapsed by default. To expand these sections always, from the main menu select Window, Preferences, Desktop Client, and select the Show expandable groups always expanded check box. This preference setting takes effect only if the Use Application-level defaults check box is cleared.

3. Complete the required fields.

4. (Optional) Perform the following actions to override the default run frequency or add a run frequency:

   a. Ensure that the Run frequency section is expanded.

   b. Clear the Use Application-level defaults check box in the Run frequency section.

   c. Click the ellipses (…) in the When field to build and test your run criteria.

      Note: If you are familiar with the scheduling terms, type your run criteria in the When field, for example, FRIDAY.

      The Run dialog opens.

   d. Select Use generated statement and choose scheduling terms from the lists that are provided. Or select Use free format statement and type your run criteria in the text box.

   e. Click Test to test your run criteria.

      The Test Results text box displays the date and time of the next ten job executions.

   f. Click OK to accept your run criteria.

      The When field displays your run criteria.

   g. Click Show in calendar.

   The calendar highlights the days when the job is selected to run based on all Run and Do not run statements for the job.

   Note: By default, this feature uses the SYSTEM calendar. If your run criteria uses terms from a calendar other than the SYSTEM calendar, verify that the Run dialog has the correct calendar specifications.

5. Complete the other optional fields as appropriate and click OK.

The details of the job are defined.

6. Repeat for all the jobs in the Application.

   The job details for all jobs in the Application are defined.

Upload the Application

You can upload the Application to the server after defining the Application. The server maintains all the versions of an Application. If you modify an Application and upload the new version to the server, it archives the current version and replaces it with the new version. Each archived version has a version number, a last modified date, an author, and an optional comment.

Follow these steps:

1. Right-click the Application in the Application Workspace view, and select Upload from the pop-up menu.

   The Upload Application to Repository dialog opens.

2. Enter a comment in the Comments text box, such as the reason for the upload.
3. (Optional) Select Unlock application after upload to unlock the Application.

   The server allows other users to update and upload the Application.

4. Click OK.

   The Application is uploaded to the server. If an older version of the Application exists, the server archives the old version.

Note: To save the Application on your local computer, right-click the Application in the Application workspace, and select Save As.

More information:

Application Versions (see page 16)

Simulate the Event

You simulate Events using different run conditions to verify that the correct jobs are selected to run in the Application and that the jobs run in the correct order. If your Application includes JavaScript scripts, you can also simulate Events to check for JavaScript errors and review the resolution of symbolic and global variables that are resolved at trigger time. Before you simulate, verify that you have uploaded the latest version of your Application to the server.

Note: You can also simulate your Event from the Services perspective. In the Events view, right-click the Event and select Simulate from the pop-up menu.

Follow these steps:

1. Right-click the Event that you want to simulate in the Application Events view, and select Simulate from the pop-up menu.

   The Simulate Event dialog opens.

2. Click the button next to the Schedule criteria field to build your schedule criteria.

   Note: If you are familiar with the scheduling terms, you can type your schedule criteria, such as FRIDAYS, directly in the When field.

   The Simulate dialog opens.

3. Specify the scheduling terms using the following steps:

   a. Select an option from the Occurrences section.

      You can select a relative occurrence (1st, 2nd, 3rd, 4th, Nth, Nth last, last) or a specific date.

   b. (Optional) Select a day from the Type of Day section if you selected an occurrence other than Date.

      You can select relative days (day, weekday, workday, or holiday) or a specific day of the week. You can also select holidays and special days that are defined in your calendar.

   c. (Optional) Select a period from the Period section if you selected an occurrence other than Date.

      You can select relative periods (week, month, or year) or a specific month. You can also select a special period that is defined in your calendar.

   d. Click OK.

4. (Optional) Complete the following fields as appropriate:

   Root jobs to run

   Specifies a subset of jobs in the Application that you want to simulate.

   Note: You can separate the list of jobs to run with commas. For example, enter **A,B,C**. To select a job with all of its successors (descendants), you can append a plus sign (+). For example, enter **D+**.

User Parameters to pass to Event

Specifies the parameter fields to pass user information to the Application during the simulation.

5. Click OK.

   The simulation opens in a separate window. On the left pane, a graphical representation of the Application displays the jobs that are selected to run based on the simulation criteria you specified. On the right pane, the details of the corresponding job definitions appear.

   Note: To display the job details, from the main menu click Window, Preferences, Desktop Client, Services Perspective, Simulate, and select the View Job Details text check box.

Example: Simulate a Sample Application

Suppose that jobs A, B, C, and D run every day. On Fridays, job E also runs. On the last workday of the month, job F also runs. Depending on the day, the server selects different jobs to run and builds the Application with the selected jobs.

For this Application, you could test the job's run criteria, using the Schedule criteria field of the Simulate Event dialog, to simulate on the following days:

- A day other than Friday, such as Monday:
  - If the following Monday is not the last workday of the month, jobs A, B, C, and D appear.
  - If the following Monday is the last workday of the month, job F also appears.
- Friday:
  - If Friday is not the last workday of the month, jobs A, B, C, D, and E appear.
  - If the following Friday is the last workday of the month, job F also appears.
- The last workday of the month:
  - If the last workday of the month is not a Friday, jobs A, B, C, D, and F appear.
  - If the last workday of the month is a Friday, job E also appears.

You have successfully defined an Application. The Application can now run.

More information:

Workload Simulation (see page 496)

Locate a Job in the Application Workspace

You can locate a specific job in the Application workspace. This feature is a useful tool for locating jobs in large Applications with many jobs.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click in the Application workspace, and select Locate Job from the pop-up menu.

   The Locate Job dialog opens.

3. Enter the job name in the Job name field, and click OK.

   Note: You must include a wildcard for a partial name. For example, A* locates all jobs with names that start with A.

   A list of all of the jobs that match the criteria appear in the list box.

   Note: If one job matches the criteria, the job is highlighted in the graph and you can omit the next step.

4. Select the job in the list box, and click Search.

   The job is highlighted in the Application workspace.

Edit the Job Definitions of Multiple Jobs in an Application

You can edit the job definitions of multiple jobs at once in an Application to save time. This feature is useful for modifying the job definitions of many jobs in a large Application. For all jobs, you can modify the Qualifier, Agent, Agent group, and Run frequency fields. For example, you can set the Qualifier field to TEST for all the UNIX jobs in your Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Do *one* of the following to select the jobs that you want to modify:

   - Select the jobs that you want to modify in the graph, right-click in the workspace, and select Job Updates from the pop-up menu.
   - Right-click in the workspace, and select Jobs Search and Update from the pop-up menu.

     The Search jobs page opens.

     Enter search strings in the following fields as required and click Next:

     - Job Type— Searches by a single job type. You cannot use wildcards in this field.
     - Name—Searches by name of the job. Applies to all job types.
     - Qualifier—Searches by qualifier of the job. Applies to all job types.
     - Agent—Searches by agent. Applies to all job types except Task, Link, and External-Same Scheduler jobs.
     - Agent group—Searches by agent group. Applies to only Windows and UNIX jobs.
     - Use Application-level defaults (Agent section)—Searches for jobs that do not have an agent or agent group specified in the job definition.
     - Run frequency—Searches by run frequency. Applies to all job types. The more Run and Do Not Run statements you specify, the more restrictive the search will be. You cannot use wildcards in this field.
     - Use Application-level defaults (Run frequency section)—Searches for jobs that do not have a run frequency specified in the job definition.
     - Command to run—Searches by the command that executes when the job runs. It must be a valid path name. Applies to only UNIX, Windows, Tandem, and Open VMS jobs.
     - Arguments to pass—Searches by the arguments that are passed to the job. Applies to only UNIX, Windows, Tandem, and Open VMS jobs.

Notes:

- The search is not case-sensitive.
- You do not have to specify values for all the search fields. The more search fields you specify, the more restrictive the search will be. You do not need to specify a value in a search field to modify that field.
- You can use the ? wildcard to match a single character and the * wildcard to match one or more characters for all the search fields except Job Type and Run frequency. For example, W?N matches WIN but not WN, whereas W*N matches WN, WIN, and WIIN.

The Modify jobs page opens and lists all the selected jobs or jobs that match the search criteria.

3. Select the Modify check boxes of each parameter you want to modify and specify new values as required.

   Notes:

   - To clear a field's value, select the Modify check box for the field and leave the field empty.
   - You cannot modify the Name field.
   - You cannot modify the Qualifier field for Windows or UNIX jobs if you have selected the Run on All Agents option button.
   - You cannot modify the Agent field for Task, Link, and External-Same Scheduler jobs.
   - You can modify the Agent group field only for Windows and UNIX jobs.
   - You can modify the Command to run and Arguments to pass fields only for UNIX, Windows, Tandem, and Open VMS jobs.
   - You cannot modify a script to a command or a command to a script for a UNIX job.

4. Click Next.

   A validation for all modifications is performed.

   - If there are warnings or errors, the Problems page lists the problems found when validating the modifications. Errors must be corrected before you can continue with the modifications. Warnings can be ignored. Only modifications without warnings or errors will be performed.
   - If there are no warnings or errors, the Preview updates dialog opens and lists all the modifications that will be performed on the selected jobs in the Changes to be performed section. You can also compare the original values of the job's parameters with the updated values in the Compare Job Modifications section.

   Note: In the Changes to be performed section, you must clear the check boxes beside the jobs you do not want to apply the changes to.

5. Click Finish to apply the selected changes to all the selected jobs.

The job definitions of all the selected jobs are modified.

Example: Modify the Agent for Multiple Jobs in an Application at Once

Suppose that you have an Application named APP01 that contains a few Windows jobs. You want to modify the agent they run on to AGENT_B120.

To modify the agent for multiple jobs in an Application at once

1. Open the APP01 Application in the Define perspective.
2. Right-click in the Application workspace, and select Jobs Search and Update from the pop-up menu.

   The Search jobs dialog opens.
3. Select Windows from the Job type drop-down list, and click Next.

   The Modify jobs dialog opens.
4. Select the Modify check box in the Agent section.
5. Ensure that the Use Application-level defaults check box is cleared and the Agent option button is selected.
6. Select AGENT_B120 from the Agent drop-down list, and click Next.

   The Preview updates dialog opens.
7. Verify the changes in the Preview updates dialog and click Finish.

   The job definitions of the Windows jobs are modified.

Download an Application

You can download the Application's current version or an older version if required.

Follow these steps:

1. Open the Define perspective.

   Your server connection appears in the Application Workspace view.

2. Right-click the server connection, and select Download from the pop-up menu.

   The Download Application(s) from Repository dialog opens.

3. Enter a search string in the Search string field.

   Note: For example, if your Application starts with JB, type JB to show a list of Applications that begin with JB. You can also use a wildcard ? to match a single character and a wildcard * to match zero or more characters. For example, J?C matches JBC but not JC, whereas J*C matches JC, JBC, and JBBC.

   The Applications that match your search string are listed.

4. Select the Applications you want to download.

   Note: To download an archived version, click Archived Versions and select the archived versions you want to download in the Archived Versions section.

5. (Optional) Select Lock application(s) after download to lock the Application.

   Other users (with different user IDs) are prevented from updating and uploading the Application until you manually unlock the Application.

   Note: Only one version of an Application can be locked at a time.

6. Click OK.

   The selected Application versions are downloaded.

More information:

Application Versions (see page 16)

Delete an Application

By default, when you delete an Application, the server deletes the current and all archived versions of the Application. You can also delete all or selected archived versions.

Follow these steps:

1. Open the Define perspective.

   Your server connection appears in the Application Workspace view.

2. Right-click the server connection, and select Delete from the pop-up menu.

   Note: If you have downloaded the Application, right-click the Application and select Delete to delete the Application.

   The Delete Application(s) from Repository dialog opens.

3. Enter a search string in the Search string field.

   Note: For example, if your Application starts with JB, type JB to show a list of Applications that begin with JB. You can also use a wildcard ? to match a single character and a wildcard * to match zero or more characters. For example, J?C matches JBC but not JC, whereas J*C matches JC, JBC, and JBBC.

   The Applications that match your search string are listed.

4. Select the Applications you want to delete, and click OK.

   Note: To delete an archived version, click Archived Versions and select the archived versions you want to delete in the Archived Versions section or select Delete only archived versions of selected applications to delete all archived versions.

   If the Application being deleted is associated with any Events, the Events Affected dialog opens and lists the Events that are affected by the deletion. Otherwise, a confirmation dialog appears and you can skip the next step.

   Note: If you selected multiple Applications, the Affected Artifacts dialog lists all of the Events that are affected by the deletion of the selected Applications.

5. Select one of the following, if applicable:

   - Yes—Proceeds with the deletion.

     Note: Events listed in the table may not trigger.

   - No—Cancels the deletion request.

     Note: To avoid the warning, you can update the Application specified in the Event definition.

6. Click Yes in the confirmation dialog, if applicable.

   The selected versions of the Application are deleted.

More information:

Application Versions (see page 16)

Lock an Application

To avoid collisions caused by concurrent updates to an Application, you can manually lock the Application.

Note: The server locks the current version of the Application. You cannot lock old Application versions.

Follow these steps:

1. Open the Define perspective.

   Your server connection appears in the Application Workspace view.

2. Right-click the server connection, and select Lock from the pop-up menu.

   Note: If you have downloaded the Application, right-click the Application and select Lock to lock the Application.

   The Lock Application(s) from Repository dialog opens.

3. Enter a search string in the Search string field.

   Note: For example, if your Application starts with JB, type JB to show a list of Applications that begin with JB. You can also use a wildcard ? to match a single character and a wildcard * to match zero or more characters. For example, J?C matches JBC but not JC, whereas J*C matches JC, JBC, and JBBC.

   The Applications that match your search string are listed.

4. Select the Applications you want to lock and click OK.

   Note: Press the Ctrl key while making your selections to select multiple Applications at once.

5. Click OK.

   The selected Applications are locked if they are currently unlocked.

Unlock an Application

You can unlock a locked Application to let other users update and upload the Application.

Follow these steps:

1. Open the Define perspective.

   Your server connection appears in the Application Workspace view.

2. Right-click the server connection, and select Unlock from the pop-up menu.

   Note: If you have downloaded the Application, right-click the Application and select Unlock to unlock the Application.

   The Unlock Application(s) from Repository dialog opens.

3. Enter a search string in the Search string field.

   Note: For example, if your Application starts with JB, type JB to show a list of Applications that begin with JB. You can also use a wildcard ? to match a single character and a wildcard * to match zero or more characters. For example, J?C matches JBC but not JC, whereas J*C matches JC, JBC, and JBBC.

   The Applications that match your search string are listed.

4. Select the Applications you want to unlock and click OK.

   Note: Press the Ctrl key while making your selections to select multiple Applications at once.

5. Click OK.

   The selected Applications are unlocked if they are currently locked.

Set Application Defaults

You can set Application defaults to save time entering common job or Application properties in the Applications. Application defaults are characteristics common to all the Applications you define. You can override Application defaults for a specific Application in the Application properties dialog.

Note: Specify Application defaults prior to creating a new Application.

Follow these steps:

1. Select Window, Preferences from the main menu.

   The Preferences dialog opens.

2. Click Desktop Client, Define Perspective, Application defaults in the left pane.

   The Application defaults page opens in the right pane.

3. Specify the Application defaults and click OK.

   The Application defaults are set.

Example: Set Up a Global Email Notification

You can set up a global email notification that sends an email to an operator whenever a job in any Application fails.

To set up a global email notification

1. Select Window, Preferences from the main menu.

   The Preferences dialog opens.

2. Click Desktop Client, Define Perspective, Application defaults, Notifications in the left pane.

   The Notifications page opens in the right pane.

3. Select the Email tab and click New.

   The New Email Notification dialog opens.

4. Select Fail in the Monitor states section.

5. Enter the operator's email address in the To field and click Add.

6. Enter an email subject, such as **Job has failed**, in the Subject field and click OK twice.

   The server sends an email notification to the operator whenever a job fails in an Application.

Add Comments to an Application

You can add comments to an Application to store information about the Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Enter your comments in the Comments text box and click OK.

   The comments are added to the Application.

Delay Application Generations from Running Until the Current Generation Completes

You can prevent Application generations from running concurrently by delaying a new generation from running until the current generation completes. If a previous generation of that Application is running, the server puts the new Application generation in an APPLWAIT state. When the current generation completes, the server removes the APPLWAIT state on the new generation, and the new generation runs.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Wait for previous generation check box and click OK.

   The server delays the new generation from running until the current generation completes.

More information:

Concurrent Application Generations (see page 18)

Prevent Events from Triggering While the Current Generation is Running

You can prevent Application generations from running concurrently by preventing Events associated with the Application from triggering while the current generation is running.

Note: An Application can be triggered by multiple Events, each with its own scheduling criteria. If you want to prevent a specific Event associated with the Application from triggering while the current generation (triggered by the same Event) is running, you can use a similar option at the Event-level.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Do not trigger if active check box and click OK.

   If a generation of the Application is running and the next scheduled execution of its associated Event occurs, the server does not trigger the Event, causing the Event to miss its scheduled execution time. The Event triggers at its next scheduled time when the current generation completes.

More information:

Concurrent Application Generations (see page 18)

Delay Each Job in an Application from Running Until the Job Completes in a Previous Application Generation

You can prevent jobs in an Application from running concurrently in two or more Application generations by delaying each job in the Application from running until the job completes in a previous generation of the Application. If a job is running in a previous Application generation, the server puts the job in the new Application generation in a JANCWAIT state. When the job completes in the previous Application generation, the server removes the JANCWAIT state on the job in the new Application generation, and submits the job when its dependencies are met.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Do *one* of the following:

   - If you want each job in the Application to wait for the same job to complete only in the previous Application generation, select the The last generation option from the Wait if the job with the same name is not complete in drop-down list.
   - If you want each job in the Application to wait for the same job to complete in all previous Application generations, select the Any previous generation option from the Wait if the job with the same name is not complete in drop-down list.

   The server delays each job in an Application from running until the same job completes in a previous Application generation.

More information:

Concurrent Application Generations (see page 18)

Enable Anticipated End Times and Critical Path Analysis

By default, the server does not calculate anticipated end times or do critical path analysis. You can enable anticipated end times and critical path analysis in the Application properties.

Note: If the Propagate dueout time option is enabled in the Application properties, the server automatically enables anticipated end times and critical path analysis.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Estimate end time check box and click OK.

   The server calculates anticipated end times and does critical path analysis when the Application runs.

   Note: If none of the jobs in the Application have an average execution time when the Application runs, the server disables anticipated end time and critical path analysis. In future runs, the server calculates anticipated end time and does critical path analysis based on the historical execution times of each job in the Application.

Note: If you want this option to apply to all Applications that you define, from the main menu click Window, Preferences, Desktop Client, Define Perspective, Application defaults, and select the Estimate end time check box.

Suppress Event Trigger Notifications When No Work is Selected

By default, the server sends a notification when an Event triggers and no jobs are selected to run in the Application. In an Application, you can suppress the notification that no work was selected to run. For example, a daily Application may run different workload depending on the day it is run. On holidays and special days, no jobs are selected to run, and the administrator does not need to receive an email that no jobs were selected.

Note: The notification can be through email or SNMP, and is configurable through the shared server parameters in the Admin perspective. For more information about the shared server parameters, see the *Admin Perspective Help*.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Suppress notification when no work selected check box and click OK.

   If an Event in the Application triggers and no jobs in the Application are selected to run, the server logs the "No work selected" exception, but does not send a notification.

Set the Reason Option for Job Commands as Mandatory

When a user issues a command against a job in the Monitor Perspective, the user can provide a reason for issuing the command. By default, the command reason is optional. However, you can force the user to enter the command reason by making it mandatory for a specific Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Require reason for job commands check box and click OK.

   The reason option is set for the jobs as a mandatory field. In the Monitor perspective, when a user issues a command against an Application, a subApplication, or a job within the Application they must provide a reason for issuing the command.

   Note: Some job commands do not support the reason option.

Export Application Parameters

If you are an administrator user of third-party software, you can define Applications using CA Workload Automation DE and export the parameters to an XML file. Users of the software can use these parameters to interface with the product.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Export Application Parameters from the pop-up menu.

   The Export Application Parameter dialog opens and lists the parameters extracted from the Application definition.

3. (Optional) Type in descriptions for all of the parameters if you are exporting them for the first time, or alternatively do the following:

   - Click Load Descriptions from Existing File to load the parameter descriptions that you specified during the previous export. Browse and select the XML file that contains the descriptions.
   - Type in descriptions for only the new parameters that were not included in the previous export.

4. Click Export as XML.

   The Save As dialog opens.

5. Choose the default name or enter a name for the XML file and click Save.

   By default, the name of the XML file is *application_name*_params.xml, where *application_name* is the name of the Application.

   The Application parameters are exported.

More information:

Specifying Application Parameters (see page 22)

Map an Application to a Business Service

You can map an Application to one or more business services defined in a CA product such as CA Spectrum SA. To integrate an Application with another CA product, you specify Application integration tags in the Application definition.

Note: To integrate CA Workload Automation DE with CA Spectrum SA, the CA Workload Automation DE connector must be installed. For more information about the connector, see the *CA Workload Automation DE Connector Guide* packaged with the connector.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click Product Integration in the left pane.

   The Product Integration page opens in the right pane.

4. Click Add.

   A new row is added to the table.

5. Specify the Application integration tag in the Integration Value column using the following format:

   `integration_type:integration_value`

   integration_type

   Specifies the integration type. Currently, only **ssa** is supported.

   integration_value

   Specifies the integration value. For integration with CA Spectrum SA, specify the name of a business service as defined in that product.

   Note: You can also specify the Application integration tag using symbolic or global variables.

6. (Optional) Repeat the previous two steps to map the Application to additional business services.

7. Click OK.

   When the Application runs, the connector maps the Application generation to the specified business services defined in CA Spectrum SA.

Example: Map a Cleanup Application to Two Business Services

Suppose that you want to map a cleanup Application to the Database and ITAdmin business services defined in CA Spectrum SA. If a problem occurs during Application execution, CA Spectrum SA generates Major alerts against the associated business services to indicate service degradation.

1. Open the cleanup Application in the Define Perspective.
2. Open the Product Integration page in the Application properties dialog.
3. Click Add.

   A new row is added to the table.
4. Enter **ssa:Database** in the Integration Value column.
5. Click Add.

   A new row is added to the table.
6. Enter **ssa:ITAdmin** in the Integration Value column.
7. Click OK.

   When the cleanup Application runs, the connector maps the Application generation to the Database and ITAdmin business services defined in CA Spectrum SA.

More information:

Application Integration Tags (see page 25)

Set the Default Zoom Settings for Application Graphs

You can set the default zoom settings for Application graphs. When you open a graphical view of the Application, the zoom defaults are used. You can override the zoom defaults for a specific Application graph by zooming in and zooming out or scrolling though the graph by using the mouse wheel or the cursor.

Follow these steps:

1. Select Window, Preferences from the main menu.

   The Preferences dialog opens.

2. Click Desktop Client, Define Perspective, Application Graph in the left pane.

   The Application Graph page opens in the right pane.

3. Do *one* of the following in the Graph Fitting section:

   - To fit the graph to the window size, select the Fit to Window check box.
   - To specify a specific zoom factor, clear the Fit to Window check box, and select a value in the Use zoom factor field.

4. Click OK.

   The zoom defaults are set for the Application graphs.

   Note: Changes to the zoom defaults do not affect graphs that are open. You must close and reopen the graphs for the changes to apply.

Chapter 2: Working with Workload Objects

This section contains the following topics:

Workload Objects (see page 56)
Locate a Job on a Server (see page 73)
Specify Run Frequency in a Workload Object (see page 75)
Run a Job on an Agent Group for Load Balancing (see page 77)
Run a Job on All Agents Defined in a Group (see page 78)
Pass Arguments to a Script or Command (see page 79)
Define Job Success or Failure (see page 80)
Specify Environment Variables (see page 82)
Add Comments to a Job (see page 85)
Set Job Defaults (see page 85)
Resubmit a Failed Job Automatically (see page 87)
Define the Jobs in a SubApplication (see page 88)
Define an On-Request Job (see page 89)
Define a Conditional Job (see page 89)
Delay a Job from Running Until the Job Completes in a Previous Application Generation (see page 90)
Override a Job's Average Execution Time (see page 91)
Identify a Critical Job in an Application (see page 92)
Specify a Profile for a Job (see page 93)
Application and Web Services Jobs (see page 95)
CA WA Jobs (see page 200)
Database Jobs (see page 202)
External Jobs (see page 219)
File Transfer Jobs (see page 223)
Informatica Jobs (see page 247)
Micro Focus Jobs (see page 255)
Monitoring Jobs (see page 263)
MSSQL Server Jobs (see page 308)
Oracle E-Business Suite Jobs (see page 320)
PeopleSoft Jobs (see page 350)
Remote Execution Jobs (see page 358)
SAP Jobs (see page 362)
SNMP Jobs (see page 406)
System Jobs (see page 434)
z/OS Jobs (see page 477)

Workload Objects

An Application consists of workload objects. Workload objects represent the work to be scheduled. Workload objects include jobs that execute programs such as Windows commands, UNIX scripts, and SAP ABAPs, objects that monitor for conditions such as file activity and triggering of SAP events, and objects that run on CA Workload Automation DE such as links and tasks.

To define a workload object, you need to identify the following:

Parameters

The parameters (job details) of a workload object are the required and optional properties of that workload object. For example, Windows jobs require a command file, and UNIX jobs require a script name or command name.

Optional parameters may include user IDs, arguments, and environment variables. Most workload objects can contain time dependencies, notifications, resource requirements, and other options.

Run frequency

You specify when a workload object runs (or does not run) using schedule criteria. When an Application is generated, the run frequency determines whether the job is selected to run.

You can add a single run-frequency condition or create a list of conditions specifying when the workload object runs or does not run.

Relationships with other workload objects

A workload object may have predecessors and successors. By default, a workload object does not run until its predecessors complete successfully and its other dependencies (time and resources) are met. When a workload object completes successfully, it releases its successors, which run after all their dependencies are met.

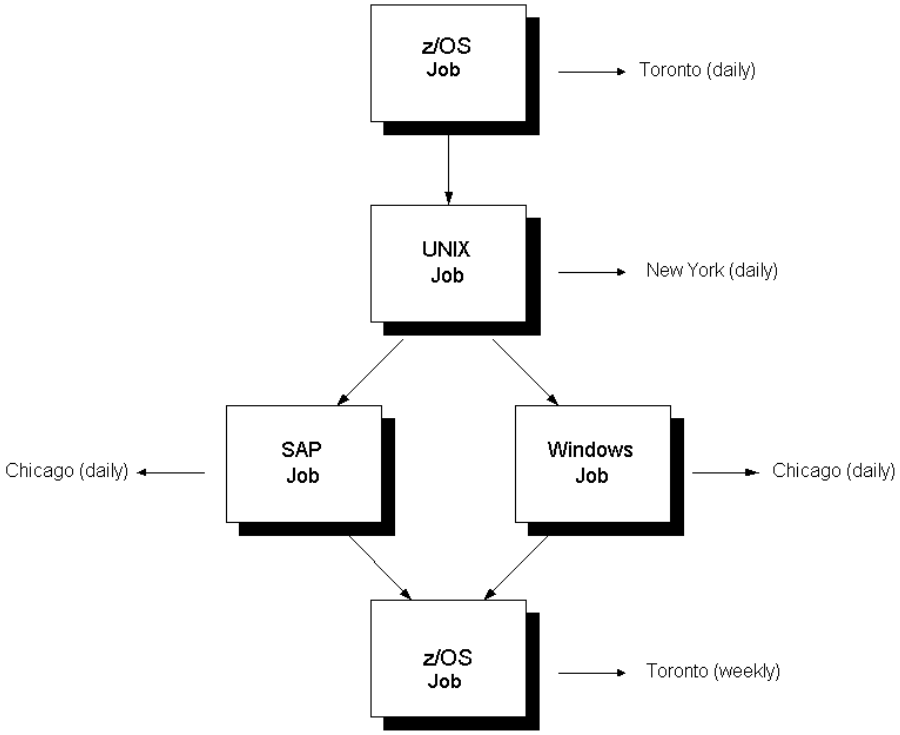
You specify a workload object's relationship with other workload objects using predecessor dependencies. You can create predecessor dependencies between jobs in the same Application or jobs in different Applications.

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.

Example: Run Workload with Different Types of Jobs

The following workload contains two z/OS jobs, a UNIX job, an SAP job, and a Windows job, running on different machines, in different locations, and at different times:



To run workload on a particular system, you need to install an agent on that system. Agents receive and respond to commands sent by the scheduling manager and transmit data and messages back to the scheduling manager.

The following table summarizes the relationship between the scheduling manager and agents:

Scheduling Manager	Agent
Is aware of the entire network	Is aware of the local environment
Sends commands and parameters to the agents	Responds to commands and parameters sent by the server
Receives data from the agents	Transmits data to the server
Makes decisions	Takes direction from the server
Schedules jobs	Runs jobs on different platforms

Note: For more information about a specific agent, see the documentation for your agent.

Agent Groups

An agent group is an explicitly defined group of Windows or UNIX/Linux agents. You need to define agent groups to use load balancing or run a job on all agents in a group.

Suppose your CA Workload Automation DE setup has ten Windows agents and ten UNIX agents; you can define the following agent groups:

- two Windows agent groups of five agents each
- two UNIX agent groups of five agents each and one Windows agent group of three agents
- one UNIX agent group of four agents, and two Windows agent groups of two agents and three agents each

These are just some of the combinations you can define; you can define any such combinations to suit your requirements, the only requirement being that you must group agents of the same operating system together.

Note: An agent group is an explicitly defined entity. For instance, all the agents of your setup do not implicitly constitute an agent group.

Load Balancing

Load balancing is the technique of balancing agent workload. Load balancing distributes workload evenly within an agent group.

To use load balancing, you need to define agent groups. When you enable load balancing on an agent group, the server allocates work to the agents in one of the following ways:

- Round robin—the server selects one agent after another of the group in a cycle; for example, if there are three agents in the group, the server allocates work to the first, second, and third agents and then starts over again with the first agent.
- Random selection—the server randomly picks an agent of the group.
- CPU usage—the server allocates work to that agent of the group that has the lowest weighted CPU usage at the time the job is submitted.

More information:

Run a Job on an Agent Group for Load Balancing (see page 77)

Running a Job on All Agents Defined in a Group

When you create a job, you can set it to run on all agents in an agent group. This feature is useful when you need to run the same job on multiple servers. For example, many companies back up their data regularly; so you can create a single backup job and set it to run on multiple agents, with each agent installed on a different server.

More information:

Run a Job on All Agents Defined in a Group (see page 78)

Run Frequency

By default, all jobs in an Application run daily. You can override this default for an individual job by specifying a different run frequency in the job definition. If you want to override this default for all the jobs in the Application, you can specify a different run frequency in the Application properties.

You cannot specify a time in the job's run frequency. When an Application is generated, the run frequency determines whether the job is selected to run. To submit a workload object at a particular time, you would specify the submission time as a time dependency in the job definition.

Note: You can specify times and frequencies (such as 4PM DAILY, EVERY 2 DAYS or 3 Times JAN 1 YEARLY) in Event definitions.

Using the Run dialog, you can define and test schedule criteria for your workload objects without memorizing the scheduling terms or the possibility of building invalid schedule criteria. With a few clicks, you can specify common schedule criteria, such as last workday of month or third last day of month.

After you become more familiar with the scheduling terms, you can save time by typing the criteria in a free form field and testing the results.

More information:

Specify Run Frequency in a Workload Object (see page 75)

On-Request Jobs

Applications can contain on-request jobs that run only when requested by a user. If the on-request job is not requested before it is eligible to run in the schedule, the server bypasses the job. You can include an on-request job in an Application to ensure that the job's predecessor dependencies are met before the job runs.

Example: On-Request Job

Jobs A, B, and C are eligible to run daily, but job B only runs if requested.

One of the following will occur:

- If Job B is not requested, when job A completes, job B is bypassed and job C runs.
- If Job B is requested, when job A completes, job B runs. When job B completes, job C runs.

More information:

Define an On-Request Job (see page 89)

Conditional Jobs

The server completes an Application after all jobs in the Application are complete or are bypassed. Applications can contain conditional jobs that may not run when selected to run.

When all non-conditional jobs in an Application complete, the server bypasses all conditional jobs, regardless of their state, and marks the entire Application complete.

Note: You can resubmit failed conditional jobs provided the Application is not complete.

Example: Optional Predecessors

By default, a job has to wait for all its predecessors to complete or be bypassed before it can run. However, one or more of the job's predecessors may be optional.

Job A and job B are conditional jobs. Job C runs when either job A or job B completes. After job C completes, you want the server to bypass job A or job B if they are not complete.

One of the following will occur:

- If job A completes before job B, job C does not wait for job B to complete. If job C completes before job B, the server marks the Application complete and bypasses job B.
- If job B completes before job A, job C does not wait for job A to complete. If job C completes before job A, the server marks the Application complete and bypasses job A.
- If job A and job B complete before job C, as soon as job C completes, all jobs are complete and the server marks the Application complete.

More information:

Define a Conditional Job (see page 89)

Job Profiling

By default, the server calculates anticipated end times and critical path using the average execution time of the last ten runs of the job regardless of execution-time differences of various runs (default profile). For more accurate average data, you can specify a job profile that considers the execution-time differences of a job for various days, times, or contexts. You can specify multiple profiles based on different criteria for a job.

For example, suppose that a job runs for one hour from Monday to Thursday and for six hours on Friday. If you do not specify a job profile, the server calculates anticipated end times and critical path based on the average execution time of the five runs of the job during the week. Using the default profile, the average execution time of the five runs is two hours, which is double the execution time for the job runs on Monday to Thursday and only one third of the execution time for the job runs on Friday. However, with job profiling, you can store job information for each run of the job based on a specified profile.

To override the default job profile, you associate a profile designation string for a given job. The profile string can be explicitly specified or calculated dynamically through JavaScript scripts and expressed using symbolic variables or the %VAR global variables function. For example, if a job execution profile uses the %APPL._SDAY symbolic variable as the profile string, the server calculates the job's anticipated end time based on the day of the week the Application is scheduled. Assuming that this job is a part of an Application that is scheduled daily, when the Application runs on a Thursday, the anticipated end time for the given job is calculated based on the job's Thursday runs and the elapsed time to execute this job is stored in the Thursday profile in the database.

You can override the job profile when you define the job or modify the job definition.

More information:

Specify a Profile for a Job (see page 93)

Using a Namespace for a User that has Different Passwords

A namespace is a name assigned when defining a user for an agent in the Topology. The namespace distinguishes the user that has access to different systems using different passwords. For example the user Bob may connect to the production system with password1 and the test system with password2. In this scenario, Bob may be set up using a namespace to identify each system he is accessing, for example, production and test.

Use the following format to define a user using a namespace:

Namespace:*user_name*

Namespace

Specifies the namespace name where the user is defined for the agent in the Topology.

User_name

Specifies the user defined in the Topology for the agent.

Example: Production:Bob, Test:Bob

Continuous Monitoring Usage

Using Alerts, you can continuously monitor for a specific condition and automatically take action when that condition occurs. For example, a File Trigger job can continuously monitor changes in a file's size and send an Alert each time the file expands or shrinks by a certain amount. Alerts let you automate your workload processing.

Note: To stop monitoring, you must force the monitoring job complete. This releases the job's successors, if any.

The following is true depending on your agent:

- CA WA Agent for UNIX, Linux, or Windows—An Alert can continuously monitor any of the following:
  - CPU usage
  - Disk space
  - File activity (created, updated, expanded, and shrunk)
  - Text strings within a file
  - Windows Event Logs
  - SNMP trap information
- CA WA Agent for Databases—An Alert can continuously monitor changes to database tables.
- CA WA Agent for SAP—An Alert can continuously monitor SAP events and processes.
- CA WA Agent for Application Services—An Alert can continuously monitor the following:
  - Messages published to a JMS topic or queue
  - Changes to MBean attributes
  - MBean notification type

To monitor continuously, you specify an Alert in the job definition. The Alert can trigger an Event or run a JavaScript script.

Note: For information about defining Alerts, see the *Services Perspective Help*.

Job Output Variables

The server creates a JavaScript variable for each return value of selected job types. The return values are available as long as the Application is active. You can use the return values of a job in the subsequent jobs of that Application.

Note: To use a job's return variables in other jobs, the agent administrator must set the agent.manager.variables.allow parameter to True in the agentparm.txt file. However, the agent administrator does not have to set this parameter for CPU Monitoring and Disk Monitoring jobs.

For example, suppose that a payroll processing application comprises three jobs: the first job calculates the employee's gross pay, the second job calculates the tax deduction, and the third job calculates the employee's contribution to charity. The second job uses the employee's gross pay returned by the first job to calculate the tax deduction. The third job uses the tax amount returned by the second job to calculate the employee contribution to charity.

The return value variable names have the following format:

APPL.*job_name_job_qualifier_return_value_name*

For example, a variable name could be APPL.agent_os_purchase_order_num, where APPL is the context name, agent is the job name, os is the qualifier, and purchase_order_num is the return value variable ID.

Notes:

- *job_name* and *job_qualifier* are in lowercase.
- Periods (.) are replaced with underscores (_) in return variable names.
- All return variable names start with APPL because variables are resolved in the Application context.

The following table lists the return value names of each job type that creates output variables:

Job type	Return value names
CPU Monitoring	counteravail counterused loadaverage1m loadaverage5m loadaverage15m
Disk Monitoring	counteravail counterused

Job type	Return value names
DB Stored Procedure (see page 68)	return <stored procedure output parameters>
SQL (see page 67)	tablerowcount <selected column names>
SNMP Value Get (see page 69)	<SNMP variable OIDs>
SNMP Value Set (see page 69)	<updated variable SNMP OID>
JMX-MBean Attribute Get	variable_jmx_mbeanattr
JMX-MBean Attribute Set	variable_jmx_mbeanattr
JMX-MBean Operation	value

Example: Return Variables from a CPU Monitoring Job

The following example shows the return variables from a CPU Monitoring job.

To return values from a CPU Monitoring job

1. Enter **CPUMonitor0** in the Name field of the Basic page.
2. Select the Return immediately option button from the Wait mode section.
3. Select the Within option button from the Range section.
4. Select the Available option button from the Monitor when CPU percentage section.
5. Click OK.

   The CPU Monitoring job return variables are as follows:

   - APPL.cpumonitor0_counteravail
   - APPL.cpumonitor0_counterused
   - APPL.cpumonitor0_loadaverage1m
   - APPL.cpumonitor0_loadaverage5m
   - APPL.cpumonitor0_loadaverage15m

Example: Return Variables from a JMX-MBean Attribute Get Job

The following example shows the return variables from a JMX-MBean Attribute Get job.

To return values from a JMX-MBean Attribtue Get job

1. Enter the following information in the Basic page:
   - Name—JMXMbeanAttrGet0
   - URL —service:jmx:rmi:///jndi/rmi://localhost:9999/server
   - MBean— config
   - Attribute— cachesize
2. Click OK.

   The JMX-MBean Attribute Get job return variables are as follows:
   - APPL. jmxmbeanattrget0_variable_jmx_mbeanattr

Note: Special considerations apply to DB Stored Procedure, SQL, SNMP Value Get, and SNMP Value Set jobs because they do not have fixed names for all return values.

SQL Jobs

The following considerations apply to SQL jobs:

- Return values are the values of selected columns and the variable representing the number of selected rows (tablerowcount).
- If the SQL job returns a single row, variables named after each selected column name (in lowercase) are returned along with the tablerowcount variable. In this case, the tablerowcount variable will have the value 1 because only one row was returned.
- If the SQL job returns multiple rows, only the tablerowcount variable will be returned. It will contain the number of returned rows.
- Return variables for an SQL job may have duplicate names. For example, assume that an SQL job selects a column named tablerowcount from a table, which is the same name as the return variable representing the number of selected rows. In this case, a JavaScript variable will not be created for the selected column named tablerowcount.
- Return variables for SQL jobs (except for tablerowcount) are also stored in a JavaScript array variable named *job_name_job_qualifier_return_variables*.

  Note: *job_name* and *job_qualifier* are in lowercase.
- Variables are stored in the array in the same order as they are returned from the database. This JavaScript array variable lets you access variables with duplicated names.

Example: Return Variables from a Select Query

The following example shows the return variables from an SQL job that runs a Select query on a database table.

Suppose that a database table named TEST1 is created as follows:

```
CREATE TABLE TEST1 (ID VARCHAR(255) NOT NULL, TABLEROWCOUNT VARCHAR(100), PRIMARY KEY (ID));
INSERT INTO TEST1 (ID, TABLEROWCOUNT) VALUES(0, 'abc');
```

An SQL job named sql0 runs the following query:

```
select id, tablerowcount from test1
```

To return variables from a select query

1. Enter the following information in the Basic page:
   - Name—sql0
   - Agent name—WLA_AGT
   - SQL command—select id, tablerowcount from test1
2. Click OK.

   The job returns the following return variables and their values:
   - APPL.sql0_tablerowcount = 1
   - APPL.sql0_id = '0'
   - APPL.sql0_return_variables[0] = '0'
   - APPL.sql0_return_variables[1] = 'abc'

Note: For JavaScript variables to be created for SQL jobs, your system requires release 11.3 or higher of the agent.

DB Stored Procedure Jobs

The following considerations apply to DB Stored Procedure jobs:

- The return variables for this job type are the stored procedure output parameters and the return value. The return variables are also stored in a JavaScript array variable named *job_name_job_qualifier*_return_variables.

  Note: *job_name* and *job_qualifier* are in lowercase.
- Variables are stored in the array in the same order as they are returned from the database.

Example: Return Variables from a Stored Procedure

The following example shows the return variables from a DB Stored Procedure job that runs a stored procedure.

Suppose that a stored procedure named test_procedure is defined as follows:

```
create procedure test_procedure(@param1 varchar(100) output, @param2 integer) as set
@param1 = 'value' return(@param2);
```

A DB Stored Procedure job named dbstoredprocedure0 runs the stored procedure.

To return variables from a stored procedure

1. Enter the following information in the Basic page:
   - Name—dbstoredprocedure0
   - Agent name—WLA_AGT
   - Procedure name—test_procedure
2. Enter the following parameters in the Parameters page:
   - param1:
   - Type—param1 out varchar, Value—abc
   - param2:
   - Type—param2 in integer, Value—123
3. Click OK.

   The job returns the following return variables and their values:
   - APPL.dbstoredprocedure0_return = 123
   - APPL.dbstoredprocedure0_param1 = 'value'
   - APPL.dbstoredprocedure0_return_variables[0] = abc
   - APPL.dbstoredprocedure0_return_variables[1] = 123

Note: For JavaScript variables to be created for DB Stored Procedure jobs, your system requires release 11.3 or higher of the agent.

SNMP Value Get and SNMP Value Set Jobs

You must note the following consideration about SNMP Value Get and SNMP Value Set jobs: the format of the returned OIDs (numeric or string) is controlled by parameters defined on the agent.

Example: Return Variables from an SNMP Value Get Job

The following example shows the return variables from an SNMP Value Get job.

To return values from an SNMP Value Get job

1. Enter the following information in the Basic page:
   - Name—SNMPValueGet0
   - MIB—cybermation.mib
   - SNMP Object ID—1.3.6.1.4.1.11203.7.1.0
   - Host name—snmp_host
   - Port—161
2. Select 2 from the Version drop-down list.
3. Enter **public** in the Community field.
4. Click OK.

   The SNMP Value Get return value variables are as follows:
   - APPL.snmpvalueget0_1_3_6_1_4_1_11203_7_1_0
   - APPL.snmpvalueget0_return_variables[0]

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 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 SNMP v2/v3 INFORM messages only)
- SNMP Value Get
- SNMP Value Set
- Web Service RPC/Encoded
- Web Service Document/Literal

By default, the serialized Java object is stored on the agent computer in the spool directory, using the job name and a numeric suffix as the file name. You can redirect the output to a destination file.

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:

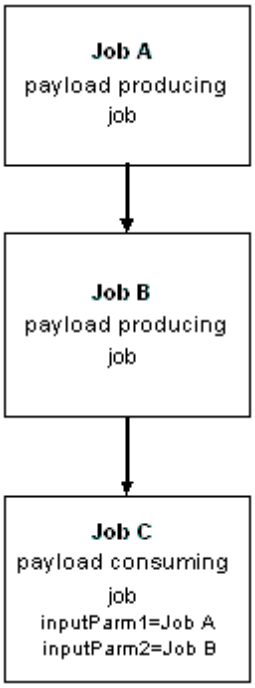
- Entity Bean
- JMS Publish
- JMX-MBean Attribute Set
- JMX-MBean Create Instance
- JMX-MBean Operation
- POJO
- RMI
- Session Bean

- SNMP Trap Send
- SNMP Value Set
- Web Service RPC/Encoded
- Web Service Document/Literal

You must define a payload consuming job and its dependent payload producing job within the same Application. The payload producing job must also be a predecessor job to the payload consuming job although it doesn't need to be an immediate predecessor.

Note: When you upload an Application that contains payload consuming jobs and their dependent jobs, CA WA Desktop Client validates the dependencies. If an error occurs in the dependency definition between one or more jobs, a Payload Dependencies Validation Errors dialog opens. The dialog lists the jobs to correct.

The following diagram illustrates a job flow in which a payload consuming job named Job C uses the output produced by payload producing jobs named Job A and Job B. In this example, Job B is dependent on Job A and Job C is defined to take the output from Job A as the value for the input parameter named inputParm1 and the output from Job B as the value for the input parameter named inputParm2.



Locate a Job on a Server

You can find a specific job uploaded as a part of an Application to the server. This feature is useful for locating jobs across Applications defined on a server.

Note: To locate a job in an Application, you must have at least Read permission to the Application.

Follow these steps:

1. Open the Define perspective.

   Your server connection appears in the Application Workspace view.

2. Right-click the server connection and select Find Job from the pop-up menu.

   The Find Job dialog opens.

3. Enter search strings in the following fields in the Filter section:

   Job Name

   Specifies the name of the job that you want to find.

   Application

   Specifies the name of the Application that the job belongs to.

   Job Qualifier

   Specifies the qualifier of the job.

   Job Type

   Specifies the type of job you are searching for.

   Agent

   Specifies the name of the agent the job runs on.

   Note: Jobs that run on an agent group are not displayed in the search results even if the agent group contains the specified agent.

Agent Group

Specifies the name of the agent group the job runs on.

Notes:

- In the search criteria fields, you can use the asterisk (*) as a wildcard for zero or more characters and the question mark (?) as a wildcard for a single character. For example, if the job name starts with WIN, type WIN* in the Job name field to show a list of jobs that begins with WIN. If the job name is exactly three characters, starts with W, and ends with N, type W?N in the Job name field. Note that W?N matches WIN but not WN, whereas W*N matches WN, WIN, and WIIN.
- You do not have to specify values for all the search fields. The more search fields you specify, the more restrictive the search will be.

4. (Optional) Select the Case sensitive check box to make your search case-sensitive.

   Note: By default, the search is not case-sensitive.

5. Click Find.

   The jobs that match your search strings are listed in a table below the Filter section.

6. Select the Application in the table that contains the job you are searching for, and click Open.

   Note: You can select multiple Applications by holding the Ctrl key while selecting the Applications.

   The selected Applications are downloaded from the server to the Application Workspace view.

Specify Run Frequency in a Workload Object

When you define a workload object, you can override the default run frequency of daily.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Ensure that the Run frequency section is expanded.

4. Clear the Use Application-level defaults check box in the Run frequency section.

5. Click the ellipses (…) in the When field to build and test your schedule criteria.

   Note: If you are familiar with the scheduling terms, you can type your schedule criteria directly in the When field, for example 3AM, HOURLY, or 10AM DAILY.

   The Run dialog opens.

6. Specify the scheduling terms using the following steps:

   a. Select one of the following options from the Occurrences section:

      - Every
      - 1st, 2nd, 3rd, 4th, Nth, Last, Nth-Nth
      - Nth last
      - Date

   b. (Optional) Select a day from the Type of Day section if you selected an occurrence other than Date.

      Note: You can select relative days (day, weekday, workday, or holiday) or a specific day of the week. You can also select holidays and special days defined in your calendar.

   c. (Optional) Select a period from the Period section if you selected an occurrence other than Every or Date.

      Note: You can select relative periods (week, month, year) or a specific month. You can also select a special period defined in your calendar.

   d. (Optional) Select the appropriate check box from the On holiday section to make adjustments to your schedule criteria for holidays.

      Note: This option does not apply to jobs that run on non-workdays such as DAILY.

7. Click Test to test your schedule criteria.

   The Test Results text box displays the date and time of the next 10 run frequencies.

8. Click OK to accept your schedule criteria.

   The Run dialog closes and the When field displays your schedule criteria.

9. (Optional) Click Add Run to add another run statement or click Add Do not run to specify an exception to the schedule criteria.

10. Click Show in calendar to see when the workload object runs.

    The calendar highlights the days when the workload object is selected to run based on all Run and Do not run statements for the job.

    Note: By default, this feature uses the SYSTEM calendar. If your run criteria uses terms from a calendar other than the SYSTEM calendar, verify that your Run dialog has the correct calendar specifications.

11. Complete the other fields as appropriate and click OK.

    The workload object runs with the frequency you specified.

Example: Run a Job on the Fourth Last Workday of February

You can run a job on the fourth last workday of February by defining the following statement:

```
Run LAST WORKDAY OF FEBRUARY LESS 3 WORKDAYS
```

Example: Run a Job on Weekdays and Sundays

You cannot specify two run conditions in the same run frequency, so Run WEEKDAYS SUN is not valid. In this case, you need to define two Run statements as follows:

```
Run WEEKDAYS
Run SUNDAYS
```

You can also specify an equivalent run frequency such as Run DAILY EXCEPT SAT.

Example: Specify Exceptions to Run Frequency

You may need to specify exceptions to run frequency. For example, you may want to run a job on workdays except Fridays. You can specify this run frequency using Run and Do not run statements as follows:

```
Run WORKDAYS
Do not run FRIDAY
```

You can also use the following statement:

```
Run WORKDAYS EXCEPT FRIDAY
```

More information:

Run Frequency (see page 59)

Run a Job on an Agent Group for Load Balancing

When you add a job to an Application, you can set it to run on an agent group for load balancing.

Note: If you select load balancing at the Application level, all jobs in the Application inherit the option. If you do not want the jobs to inherit the option, you need to uncheck the option Use Application Defaults.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Expand Agent and de-select the Use Application-level defaults check box.

   The Agent options become editable.

4. Select the Agent group option button, and from the drop-down list, select or type in the agent group you want the job to run on.

5. Select the Load balancing option button and click OK.

   The job will run on the agent group with load balancing.

More information:

Load Balancing (see page 58)

Run a Job on All Agents Defined in a Group

When you add a job to an Application, you can set it to run on all agents in an agent group.

Note: If you select Run on all agents at the Application level, all jobs in the Application inherit the option. If you do not want the jobs to inherit the option, you need to uncheck the option Use Application Defaults.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Expand Agent and de-select the Use Application-level defaults check box

   The Agent options become editable.

4. Select the Agent group option button, and from the drop-down list, select or type in the agent group you want the job to run on.

5. Select the Run on all agents option button and click OK.

   The job will run on all agents in the group.

Note: There will be an instance of the job for each agent of the agent group at runtime; all instances will have the same name but different qualifiers that are automatically set by the server. The qualifier is the name of the agent on which the job runs.

More information:

Running a Job on All Agents Defined in a Group (see page 59)

Pass Arguments to a Script or Command

When running workload, you may need to pass arguments to a script or command when that script or command is invoked. Arguments can be numeric or alphabetic strings of data.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Specify the arguments in the Arguments to pass field in the order expected in the script or command, separating each argument with a blank space.

   Note: To pass an argument containing spaces, you must enclose its value in double quotes, for example, 362 "629 630" 748.

   The argument is passed to the command or the script when the job runs.

Example: Pass Multiple Arguments with Spaces

The following argument specification passes two arguments to the pay.exe executable:

```
"c:\Pay Date\Salary.dat" "c:\Pay Date\benefits.dat"
```

Both arguments contain spaces, so each argument is enclosed in double quotes.

Example: Pass Built-in Variables

The following argument specification passes a file path as an argument with two JavaScript built-in variables, representing the Application name and generation:

```
/u1/system/results/CST%(APPL._name).%(Appl._/gen)
```

Define Job Success or Failure

You can specify which exit codes indicate job success and which exit codes indicate failure. By default, the server interprets an exit code of zero (0) as a successfully completed job and any exit code other than zero as a failed job.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Exit Codes in the left pane.

   The Exit Codes page opens in the right pane.

4. Click Add.

   A new row is added to the Exit code interpretation table.

5. Complete the following fields as required:

   Code

   Specifies a single exit code, a list of exit codes separated by commas, or a range of exit codes indicated by a hyphen.

   Limits: 0 to 9999

   Note: If you specify multiple exit codes, enter the most specific codes first followed by the more general ranges.

   Interpretation

   Indicates whether the server identifies the job as a success or a failure.

6. (Optional) Click Add to specify additional exit code specifications.

7. Click OK.

   When the job ends, the server determines the job success or failure based on the exit codes you specified.

Example: Specify Multiple Successful Exit Codes

Suppose that a job ends successfully with an exit code of 0, 1, or 2 and ends in failure with all other exit codes.

To specify multiple successful exit codes

1. Open the Exit Codes page.
2. Click Add.

   A new row is added to the Exit code interpretation table.
3. Enter **1** in the Code field and select Success from the Interpretation drop-down list.
4. Click Add.

   A new row is added to the Exit code interpretation table.
5. Enter **2** in the Code field and select Success from the Interpretation drop-down list.
6. Click OK.

   The job ends successfully with an exit code of 0, 1 or 2.

   Note: You do not have to specify 0 as a successful exit code since it is the default.

Example: Define a Specific Exit Code Followed by a General Range

Suppose that a job ends in failure with an exit code of 19 and ends successfully with all other exit codes.

To specify a specific exit code followed by a general range

1. Open the Exit Codes page.
2. Click Add.

   A new row is added to the Exit code interpretation table.
3. Enter **19** in the Code field and select Failure from the Interpretation drop-down list.
4. Click Add.

   A new row is added to the Exit code interpretation table.
5. Enter **0-9999** in the Code field and select Success from the Interpretation drop-down list.
6. Click OK.

   The job ends in failure with an exit code of 19 and ends successfully with all other exit codes.

Specify Environment Variables

You can specify environment variables to define the local environment where the script, command, or the batch file runs. You can modify existing environment variables or create your own.

Note: When passing multiple environment variables to an agent, the maximum size allowed is 4 KB.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Environment Variables in the left pane.

   The Environment Variables page opens in the right pane.

4. Click Add.

   A new row is added to the Environment variables table.

5. Complete the following fields as required:

   Name

   Defines the name of a new environment variable or specifies the name of an existing environment variable.

   Note: The agent does not support spaces in environment variable names.

   Value

   Specifies a value for the environment variable.

6. (Optional) Repeat the previous two steps to specify additional environment variables.

7. Click OK.

   The server passes the name and value of each environment variable to the script, command, or batch file when the job runs.

Example: Specify Alternative Input and Output Sources

The following table lists the input and output streams that all UNIX programs run by a shell are connected to:

Stream	Default Source	Environment Variable
Standard input stream	Keyboard	STDIN

Stream	Default Source	Environment Variable
Standard output stream	Screen	STDOUT
Standard error output stream	Screen	STDERR

To specify alternative input and output sources

1. Open the Environment Variables page.
2. Click Add.

   A new row is added to the Environment variables table.
3. Enter **STDIN** in the Name field and the full path of an alternative input stream in the Value field.
4. Click Add.

   A new row is added to the Environment variables table.
5. Enter **STDOUT** in the Name field and the full path of an alternative output stream in the Value field.
6. Click Add.

   A new row is added to the Environment variables table.
7. Enter **STDERR** in the Name field and the full path of an alternative error stream in the Value field.
8. Click OK.

   The alternative input and output sources are specified.

Example: Specify Environment Variables as Part of the Variable Value

Suppose that the output of a job is written to a file named JOB_OUTPUT located in the path specified by the MY_FILES environment variable.

For this example, enter **STDOUT** in the Name field and **$MY_FILES/JOB_OUTPUT** in the Value field.

Example: Set an Environment Variable to an Empty Value

To set an environment variable to an empty value, you can enter two quotes without spaces (“”) in the Value field.

Suppose that a job runs a script under the user ID jdoe. The script uses an environment variable named VAR1, which is set in the profile file for jdoe. The server passes VAR1 and its empty value to the script.

For this example, enter **VAR1** in the Name field and **“”** in the Value field.

Example: Set Environment Variables in a PeopleSoft Job

Suppose that you want to set environment variables in a PeopleSoft job. In this example, the location of the PeopleSoft PeopleTools executables (TOOLBIN) and path to the database drivers (DBBIN) are set.

1. Open the Environment Variables page of the PeopleSoft job definition.
2. Click Add and enter the following information in the first row of the table:
   - Name—TOOLBINSRV
   - Value—%%PS_HOME%%\bin\client\winx86

   Note: To specify the PeopleSoft home directory (%PS_HOME%), you must escape the percent signs (%) by doubling them. Otherwise, the value is treated as a symbolic variable.
3. Click Add and enter the following information in the second row of the table:
   - Name—DBBIN
   - Value—c:\test\drivers
4. Click OK.

   The environment variables are specified for the PeopleSoft job.

More information:

Environment Variables in OpenVMS (see page 448)
Environment Variables in Tandem NSK (see page 450)
Environment Variables in Windows (see page 477)
Environment Variables in UNIX (see page 461)

Add Comments to a Job

You can add comments to a job to store information about the job. For example, you may want to write a comment such as “This job generates an important file” to alert an operator that this job should not be bypassed.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Comments in the left pane.

   The Comments page opens in the right pane.

4. Enter your comments in the Comments text box and click OK.

   The comments are added to the job.

Set Job Defaults

You can set job defaults to save time entering common properties that apply to all jobs of a specific type. For example, you can set job defaults for the UNIX jobs in your Application. You can override job defaults in the job definition dialogs.

Note: Specify job defaults before adding workload objects to the workspace.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the workload object you want to set job defaults for in the Palette view, and select Job Defaults from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Complete the fields as required.

4. Click OK.

   When you add workload objects onto the workspace, the job defaults you defined for that workload object are used. To override a job default, edit the job's definition.

Example: Set Default Run Frequency for UNIX Jobs

Suppose that you have an Application that contains UNIX, SAP R/3, and Windows jobs. You want the Application to run every day. You want all your UNIX jobs to run only on weekdays, except for one UNIX job, FINAL, which you want to run only on Fridays. You want the SAP R/3 and Windows jobs to run daily.

To set up the job defaults in the Application

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Set up the default run frequency of weekdays for your UNIX jobs.

   Note: Complete this step before adding UNIX jobs to your Application.

   a. Right-click the UNIX job from the System group in the Palette view, and select Job Defaults from the pop-up menu.

      The Basic page of the UNIX dialog opens.

   b. Click Add Run in the Run frequency section, and enter **weekdays** in the When field.

   c. Click OK.

3. Override the default run frequency for the UNIX job FINAL.

   a. Select the UNIX job from the System group in the Palette view, and drag the job to the workspace.

      The UNIX icon appears on the Application workspace view.

   b. Right-click the UNIX icon, and select Edit from the pop-up menu.

      The Basic page of the UNIX dialog opens.

   c. Enter **FINAL** in the NAME field.

   d. Change **weekdays** to **Friday** in the When field in the Run frequency section.

   e. Complete the remaining fields as required and click OK.

4. Define the remaining jobs in the Application.

Note: By default, all jobs run daily, so you do not have to set job defaults for your SAP R/3 and Windows jobs unless you have changed the default run frequency in the Basic page of the Application properties dialog.

When you add UNIX jobs to the Application, they will run on weekdays, since you defined weekdays as a job default. You can override a job default by editing a job's definition, as you did with FINAL.

FINAL will run on Fridays, the other UNIX jobs will run on weekdays, and all other jobs in the Application will run daily.

Resubmit a Failed Job Automatically

If a job fails, an operator can manually resubmit the job. When you define the job, you can instruct the server to automatically resubmit a job if it fails.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Ensure that the Auto-Resubmit failed job section is expanded.

5. Complete the following fields in the Auto-Resubmit failed job section:

   Retry count

   Specifies the number of times you want to resubmit the job if it fails.

   Default: 0

   Example: If you set the retry count to 2 and the job fails, the server automatically resubmits the job. If the job fails again, the server automatically resubmits the job a second time.

   Interval

   Specifies the interval between retries in minutes.

   Default: The server resubmits the job immediately after failure

   Return codes

   Specifies which exit codes to resubmit the job on.

   Example: You can resubmit a job if it fails with an return code of 4. If the job fails with another return code, it is not resubmitted.

   Note: You can specify a single return code (such as 4), a range of return codes (such as 4-9), or a list of return codes (such as 1, 2, 8, 10-9999).

6. Click OK.

   The server automatically resubmits the job if it fails.

   Note: If the job fails, it will have an AutoResubmit condition. By default, CA WA Desktop Client displays failed jobs with an AutoResubmit condition in yellow in the Monitor perspective.

Define the Jobs in a SubApplication

To define the jobs in a subApplication, you must identify the jobs that belong to the subApplication.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click a job that belongs to the subApplication, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Ensure that the subApplication section is expanded

5. Enter the name of the subApplication the job belongs to in the Member of subApplication field.

   Note: If you have already identified a job as a member of the subApplication, you can select the subApplication from the drop-down list.

6. (Optional) Select the Wait for previous generation of subApplication check box to prevent jobs in a subApplication from running until the corresponding jobs in a previous generation complete. The server places the waiting jobs in the SUBAPPLWAIT state.

   Note: You only need to select this option for one of the jobs in the subApplication.

   The job is identified as a member of the subApplication.

Define an On-Request Job

You can define jobs to run only on request. For example, an Application may contain an ad hoc job that an operator needs to run when requested to do so.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.
3. Click General in the left pane.

   The General page opens in the right pane.
4. Select the Only on request check box and click OK.

   The job is defined as an on-request job.

More information:

On-Request Jobs (see page 60)

Define a Conditional Job

You can define conditional jobs in an Application that may not run when selected to run. For example, you can use conditional jobs to represent optional predecessors.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.
3. Click General in the left pane.

   The General page opens in the right pane.
4. Select the Conditionally check box and click OK.

   The job is defined as a conditional job.

More information:

Conditional Jobs (see page 61)

Delay a Job from Running Until the Job Completes in a Previous Application Generation

When you define a job, you can prevent it from running concurrently in two or more Application generations by delaying the job from running until the job completes in a previous generation of the Application. If a job is running in a previous Application generation, the server puts the job in the new Application generation in a JANCWAIT state. When the job completes in the previous Application generation, the server removes the JANCWAIT state on the job in the new Application generation, and submits the job when its dependencies are met.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Ensure that the Wait for previous generation section is expanded

5. Do one of the following:

   - If you want the job to wait for the same job to complete only in the previous Application generation, select the The last generation option from the Wait if the job with the same name is not complete in drop-down list.
   - If you want the job to wait for the same job to complete in all previous Application generations, select the Any previous generation option from the Wait if the job with the same name is not complete in drop-down list.

   The server delays the job in the new generation from running until the same job completes in a previous Application generation.

More information:

Concurrent Application Generations (see page 18)

Override a Job's Average Execution Time

If you do not want the server to use average execution times when calculating anticipated end times, you can specify a job duration. For example, you can specify that a job takes twice as long as the average execution time to execute on Fridays or on the last day of the month.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Ensure that the Estimate end time section is expanded.

5. Enter a numerical duration in the Duration in minutes field.

   Note: If the field is disabled, select the Estimate end time check box in the Application properties.

6. Click OK.

   The server overrides the job's default average execution time.

Example: Override a Job's Average Execution Time on Fridays

On Fridays, a job takes twice as long to execute as its average execution time.

Enter the %IF statement for the Duration in minutes field as follows:

```
%IF(today('Friday'),'%WOB._avgruntime*2')
```

The today JavaScript built-in function returns true if today is Friday; otherwise, it returns false.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

More information:

Anticipated End Times (see page 19)

Identify a Critical Job in an Application

If the Application has Estimate end time enabled, you can identify one or more of its jobs as critical.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Select the Critical job check box.

   Note: If the field is disabled, select the Estimate end time check box in the Basic page of the Application properties dialog.

5. Click OK.

   The job is defined as a critical job.

Specify a Profile for a Job

By default, the server calculates anticipated end times and critical path using the average execution time of the last ten runs of the job regardless of execution time differences of various runs (default profile). For more accurate average data, you can specify a job profile that considers the execution time differences of a job for various days, times, or contexts.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click General in the left pane.

   The General page opens in the right pane.

4. Enter a profile name in the Execution profile field.

   Limits: Up to 125 characters

   Default: Default profile (average execution time of the last 10 runs)

   Notes:

   - The profile name is not case-sensitive. It is stored in the database in upper case.
   - Symbolic variables are allowed.

5. Click OK.

   The execution profile is specified for the job.

Example: Specify Job Profile for the Last Workday of the Month

Suppose that you want to create a profile named LAST that stores data for job runs on the last workday of the month.

To specify a job profile for the last workday of the month

1. Define a UNIX job and enter the following in the Script/command name field:

   ```
   \bin\sleep
   ```

2. Open the JavaScripts page of the job definition, and define the following JavaScript script in the Define or Import JavaScript to run text box in the At Event trigger time section:

   ```
   if(today('last workday of the month'))
   {
   APPL.profile='LAST';
   }
   ```

3. Open the General page of the job definition, and enter the following statement in the Execution profile field in the Estimate end time section:

   ```
   %APPL.profile
   ```

4. Click OK.

   The server uses the LAST profile for anticipated end time and critical path calculations if today is the last workday of the month.

Example: Specify a Profile for Monday's Job Runs

On Mondays, suppose that a job takes twice as long to execute as its average execution time on other days. You can create a JavaScript script that applies a different profile depending on the day of the week.

To specify a job profile for Monday's job runs

1. Define a UNIX job and enter the following in the Script/command name field:

   ```
   \bin\sleep
   ```

2. Open the JavaScripts page of the job definition, and define the following JavaScript script in the Define or Import JavaScript to run text box in the At Event trigger time section:

   ```
   if(today('monday'))
   {
   APPL.profile='MON';
   }
   else
   {
   APPL.profile='GEN';
   }
   ```

3. Open the General page of the job definition, and enter the following statement in the Execution profile field in the Estimate end time section:

   ```
   %APPL.profile
   ```

4. Click OK.

   The server uses the MON profile for anticipated end time and critical path calculations if today is Monday; otherwise, it uses the GEN profile.

More information:

Job Profiling (see page 62)

Application and Web Services Jobs

Application and Web Service jobs let you manage entity beans, session beans, and MBeans, publish and consume JMS messages, invoke programs over HTTP, call an operation within a web service, and run other types of Java-based workload.

Entity Bean

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

HTTP

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

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

JMS Subscribe

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

JMX-MBean Attribute Get

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

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

JMX-MBean Create Instance

Lets you create an MBean on a JMX server.

JMX-MBean Operation

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

JMX-MBean Remove Instance

Lets you remove an MBean from a JMX server.

JMX-MBean Subscribe

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

POJO

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

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

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.

Web Service RPC/Encoded

Lets you call an operation within a web service and pass parameters to the operation using RPC/encoded style binding. The parameters can be actual values or a serialized Java object passed by another job.

Web Service Document/Literal

Lets you call an operation within a web service and pass parameters to the operation using document/literal style binding. The parameters represent a flattened view of the XML document the agent constructs. The values passed into the XML document can be literal values or a serialized Java object passed by another job.

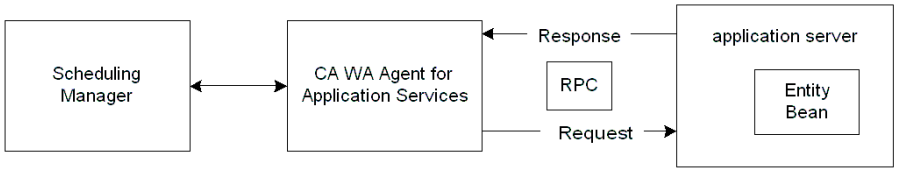
Entity Bean Jobs

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)

Create an Entity Bean Using an Entity Bean Job

You can define an Entity Bean job to create an entity bean that is stored in a relational database on your application server.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Entity Bean job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Entity Bean icon appears on the Application workspace view.

3. Right-click the Entity Bean icon, and select Edit from the pop-up menu.

   The Basic page of the Entity Bean dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that lets you access entity beans on an application server.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Initial context 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 the application can use to connect to the application server.

Example: weblogic.jndi.WLInitialContextFactory

Provider URL

Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

Example: t3://localhost:7001

Bean name

Specifies the Session Bean Java Naming and Directory Interface (JNDI) name.

Example: test.HelloJndi

Operation type

Specifies the operation to perform on the Entity Bean: CREATE, UPDATE, or DELETE.

Note: Select CREATE to define and store an entity bean instance.

5. (Optional) Specify the following additional information:

Create method name

Specifies the name of the create method. The name must always begin with create. This field is only required when the operation type is CREATE.

Default: create

Example: createaccount

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click Create Parameters in the left pane.

   The Create Parameters page opens in the right pane.

7. Click New to define the parameters.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The Entity Bean job is defined.

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.

To create an entity bean

1. Enter the following required information in the Basic page:
   - Name—CREATE
   - Agent name—APPAGENT
   - Initial context factory—weblogic.jndi.WLInitialContextFactory
   - Provider URL—t3://localhost:7001
   - Bean name—customer
   - Create method name—createcustomer
   - Operation type—CREATE
2. Add the following two value parameters in the Create Parameters page:
   - Parameter Type—String
   - Parameter Value—customerid
   - Parameter Type—String
   - Parameter Value—800-123-1234
3. Click OK.

   When the job runs, the entity bean instance is created.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Entity Bean Jobs (see page 97)
Using a Namespace for a User that has Different Passwords (see page 63)

Update an Entity Bean Using an Entity Bean Job

You can define an Entity Bean job to update the property values of an entity bean instance in a relational database on your application server.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Entity Bean job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Entity Bean icon appears on the Application workspace view.

3. Double-click the Entity Bean icon, and select Edit from the pop-up menu.

   The Basic page of the Entity Bean dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access entity beans on an application server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Bean name

Specifies the Session Bean Java Naming and Directory Interface (JNDI) name.

Example: test.HelloJndi

Operation type

Specifies the operation to perform on the Entity Bean: CREATE, UPDATE, or DELETE.

Note: Select UPDATE to modify the parameters for the entity bean instance.

Method name

Specifies the method to be invoked on the application server.

Example: hello

Finder name

Specifies the finder method name. This field is only required when the operation type is **UPDATE** or **REMOVE**.

5. (Optional) Specify the following optional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click Finder Parameters in the left pane.

   The Finder Parameters page opens in the right pane.

7. Click New to define the parameters.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click Modify Parameters in the left pane.

    The Modify Parameters page opens in the right pane.

12. Repeat Steps 7 through 10 to specify modify parameters.

13. Click OK.

    The Entity Bean job is defined.

Example: Update an Entity Bean

Suppose that you want to update the phone number for the acme company to 800-123-4567. The customer entity bean stores the customer id and phone number. The primary key for the customer is the customer id.

To update an entity bean

1. Enter the following required information in the Basic page:
   - Name—UPDATE
   - Agent name—APPAGENT
   - Initial context factory—weblogic.jndi.WLInitialContextFactory
   - Provider URL—t3://localhost:7001
   - Bean name—customer
   - Operation type—UPDATE
   - Method name—changephone
   - Finder name—acme
2. Add the following value parameter in the Finder Parameters page:
   - Parameter Type—String
   - Parameter Value—customerid
3. Add the following value parameter in the Method Parameters page:
   - Parameter Type—String
   - Parameter Value—800-123-4567
4. Click OK.

   When the job runs, the acme company's phone number is changed.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Entity Bean Jobs (see page 97)
Using a Namespace for a User that has Different Passwords (see page 63)

Remove an Entity Bean Using an Entity Bean Job

You can define an Entity Bean job to remove an instance of an entity bean stored in a relational database on your application server.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Entity Bean job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Entity Bean icon appears on the Application workspace view.

3. Right-click the Entity Bean icon, and select Edit from the pop-up menu.

   The Basic page of the Entity Bean dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access entity beans on an application server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Initial context 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 the application can use to connect to the application server.

Example: weblogic.jndi.WLInitialContextFactory

Provider URL

Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

Example: t3://localhost:7001

Bean name

Specifies the Session Bean Java Naming and Directory Interface (JNDI) name.

Example: test.HelloJndi

Operation type

Specifies the operation to perform on the Entity Bean: CREATE, UPDATE, or DELETE.

Note: Select REMOVE to remove an instance of the entity bean.

Finder name

Specifies the finder method name. This field is only required when the operation type is **UPDATE** or **REMOVE**.

5. (Optional) Specify the following optional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click Finder Parameters in the left pane.

   The Finder Parameters page opens in the right pane.

7. Click New to define the parameters.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The Entity Bean job is defined.

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.

To remove an entity bean

1. Enter the following required information in the Basic page:
   - Name—REMOVE
   - Agent name—APPAGENT
   - Initial context factory—weblogic.jndi.WLInitialContextFactory
   - Provider URL—t3://localhost:7001
   - Bean name—customer
   - Operation type—REMOVE
   - Finder name—acme
2. Add the following (type) parameter in the Finder Parameters page:
   - Parameter Type—String
   - Parameter Value—customerid
3. Click OK.

   When the job runs, the row in the customer table that corresponds to the acme customer id is removed.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Entity Bean Jobs (see page 97)
Using a Namespace for a User that has Different Passwords (see page 63)

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*.

Define an HTTP Job

You can define an HTTP job to invoke a program over HTTP. If the host you are connecting to requires user authentication, you must complete the Connection Information page.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the HTTP job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The HTTP icon appears on the Application workspace view.

3. Right-click the HTTP icon, and select Edit from the pop-up menu.

   The Basic page of the HTTP dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you invoke a program over HTTP.

   Invocation type

   Indicates the HTTP method type, which can be either GET or POST.

   Default: POST

URL

Specifies the host where the program you want to invoke resides. The URL has the following format:

http://*host*:*port*/*action*

- *host* is the name of the computer running the application server.
- *port* is the port the host uses to listen for HTTP requests. The default port is 80.
- (optional) *action* is the name of the program or servlet to be invoked. If you omit the action, you must enter a value in the Action field.

Example: http://localhost/cgi-bin/test.sh

Note: HTTP and HTTPS are supported.

5. (Optional) Specify the following additional information:

Action

Specifies the path to the servlet to be invoked.

Example: /cgi-bin/test.sh

Note: If you omit this field, you must specify the action in the URL field.

Filter

Defines the regular expression to use as a filter. The job spool file will contain lines from the HTTP response that match the filter.

6. Click Action Parameters in the left pane.

The Action Parameters page opens in the right pane.

7. Complete the following fields for each parameter.

Parameter Type

Specifies the variable that the agent passes to the program the job invokes.

Parameter Value

Defines a value for the variable specified in the Parameter Type field.

Note: Click Add to enter a parameter type and parameter value pair.

8. Click Connection Information in the left pane if connection authentication is required.

   The Connection Information page opens in the right pane.

9. Complete the following fields that are required by a website with connection authentication. Speak to your agent administrator for details.

   User

   Specifies the user name required by a website for connection authentication. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Limits: Up to 128 characters

   Examples: Bob, Production:Bob

   Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

   Origin

   Specifies the origin host name for NTLM connection authentication. If unspecified, the origin defaults to the computer name where the agent is running.

   Limits: Up to 128 characters

   Domain

   Specifies the domain for NTLM connection authentication.

   Limits: Up to 128 characters

   Don't use global proxy defaults

   Ignores the global proxy configuration specified by the proxy parameters in the agentparm.txt file. Select this option if you want to ignore the proxy for this job only. For example, you might ignore the proxy if the request is going to a server on the LAN that does not require the proxy.

Authentication order

Specifies a list of protocols that is used by a web server for authentication.

Valid values are as follows:

- BASIC
- DIGEST
- NTLM

Notes:

- You can specify any of the following protocols in any order: BASIC, DIGEST, and NTLM. To specify more than one protocol, separate the protocols by commas.

If you are connecting to an old web server that cannot negotiate authentication protocols, enter the following list in the specified order: BASIC, DIGEST, NTLM.

10. Click OK.

    The HTTP job is defined.

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.

To define an HTTP job to perform a Google search

1. Enter the following required information in the Basic page:
   - Name—google
   - Agent name—AGENT
   - Invocation type—GET
   - URL—http://google.com/search
2. Enter **.*AutoSys.*** in the Filter field.
3. Add the following parameter in the Action Parameters page:
   - Parameter Type—q
   - Parameter Value—ca workload automation
4. Click OK.

   When the job runs, the job spool file will contain all matches that contain the filter AutoSys.

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.

To define an HTTP job to subscribe to a mailing list

1. Enter the following required information in the Basic page:
   - Name—LIST
   - Agent name—AGENT
   - Invocation type—GET
   - URL—http://localhost:8080
   - Action—/examples/servlets/servlet/TheServlet
2. Add the following parameters in the Action Parameters page:
   - Parameter Type—string
   - Parameter Value—subscribe
   - Parameter Type—string
   - Parameter Value—test@abc.com
3. Click OK.

More information:

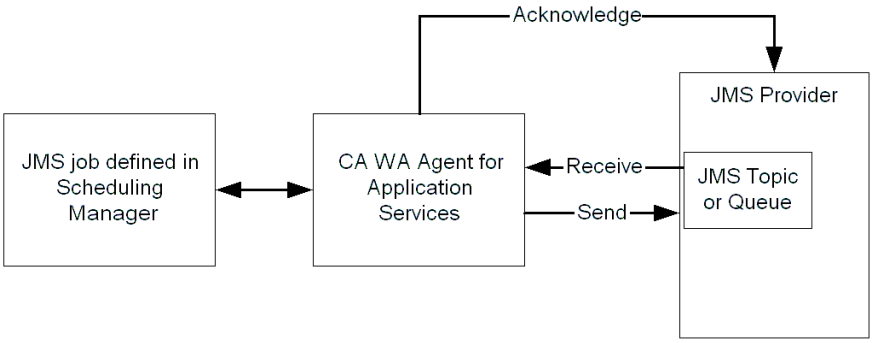
Payload Producing and Payload Consuming Jobs (see page 71)
Configure the Client for a Proxy (see page 176)
HTTP Jobs (see page 111)
Using a Namespace for a User that has Different Passwords (see page 63)

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.

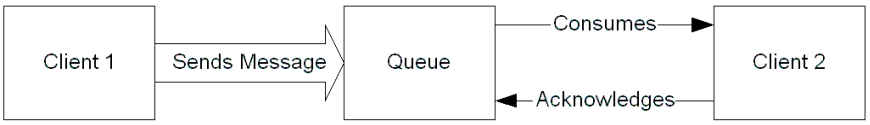
In a JMS system, a messaging server known as the JMS provider acts between two JMS clients (the publisher and the subscriber). Publishers send messages to the JMS provider while subscribers receive messages from the JMS provider.

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



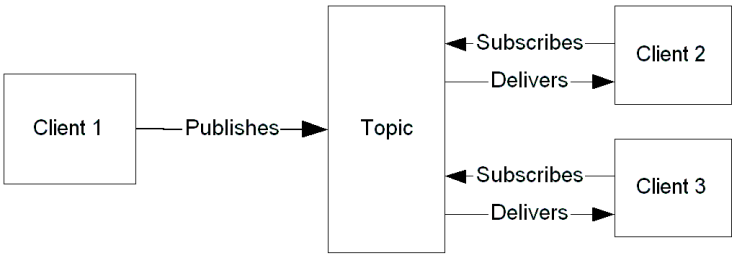
A queue is an object on the JMS server that holds messages sent by a client that are waiting to be consumed by another client. The queue retains a message until the message is consumed or the message expires.

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.

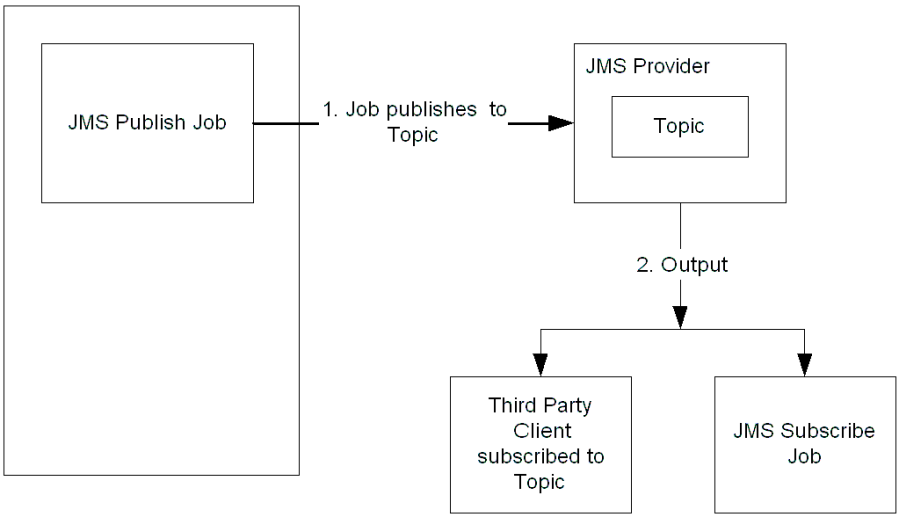
The following diagram shows two subscribers, Client 2 and Client 3, subscribed to a topic that the publisher, Client 1, publishes to:



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.

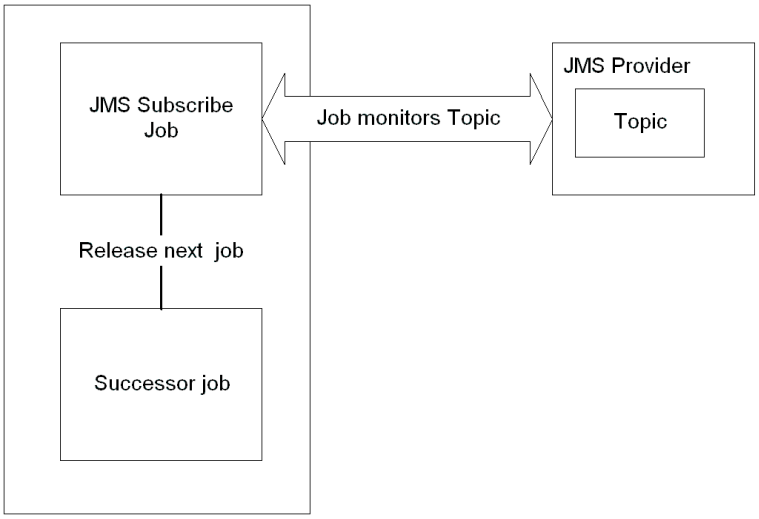
The following diagram shows a JMS Publish job scenario:



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

Define a JMS Publish Job

You can define a JMS Publish 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMS Publish job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMS Publish icon appears on the Application workspace view.

3. Right-click the JMS Publish icon, and select Edit from the pop-up menu.

   The Basic page of the JMS Publish dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that runs the JMS Publish or JMS Subscribe job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Connection 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.

Example: ConnectionFactory

JNDI destination

Specifies the Topic or Queue JNDI name. The job uses the JNDI name to indicate the destination where messages are received.

Example: MyJMSQueue

Message class

Specifies the JMS message Java class.

Example: String

Destination type

Specifies whether the job sends to a queue or publishes to a topic.

Example: TOPIC

5. (Optional) Specify the following additional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click Message Parameters in the left pane.

   The Message Parameters page opens in the right pane.

7. Click New to add message parameters for the message class.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value
   - Payload producing job—Identifies the name of a predecessor job
   - Array—Identifies more than one value

   The fields related to the option appear.

9. Complete the following fields that apply to the parameter option:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The JMS Publish job is defined.

Example: Publish a Message to the WebSphere Application Server

This example publishes the message "this is my message" to the queue named Queue. 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.

To publish a message to the WebSphere application server

1. Enter the following required information in the Basic page:
   - Agent name—AGENT
   - Initial context factory—com.ibm.websphere.naming.WsnInitialContextFactory
   - Provider URL—iiop://172.24.0.0:2809
   - Connection factory—ConnectionFactory
   - JNDI destination—Queue
   - Message class—String
   - Destination type—QUEUE
2. Add the following parameter in the Message Parameters page:
   - Parameter Type—java.lang.String
   - Parameter Value—this is my message
3. Click OK.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
JMS Publish and JMS Subscribe Jobs (see page 117)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a JMS Subscribe Job

You can define a JMS Subscribe 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMS Subscribe job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMS Subscribe icon appears on the Application workspace view.

3. Right-click the JMS Subscribe icon, and select Edit from the pop-up menu.

   The Basic page of the JMS Subscribe dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that runs the JMS Publish or JMS Subscribe job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Connection 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.

Example: ConnectionFactory

JNDI destination

Specifies the Topic or Queue JNDI name. The job uses the JNDI name to indicate the destination where messages are received.

Example: MyJMSQueue

Message class

Specifies the JMS message Java class.

Example: String

Destination type

Specifies whether the job sends to a queue or publishes to a topic.

Example: TOPIC

5. (Optional) Specify the following additional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Monitor continuously using Alert

Specifies the name of the Alert that the server triggers when a message meets the criteria specified in the Filter field.

Note: If you specify an Alert, the job monitors the topic or queue continuously, and the CA Workload Automation DE server sends the Alert each time a message matches the filter criteria.

Filter

Defines the filter used to monitor the topic or queue using regular expression logic.

Example: .*spool.*

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. Click OK.

The JMS Subscribe job is defined.

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.

To monitor a queue on a WebLogic Application server

1. Enter the following required information in the Basic page:
   - Agent name—AGENT
   - Initial context factory—weblogic.jndi.WLInitialContextFactory
   - Provider URL—t3://172.24.0.0:7001
   - Connection factory—ConnectionFactory
   - JNDI destination—Queue
   - Monitor continuously using Alert—a13
   - Filter—abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+
   - Destination type—QUEUE
   - Output destination—/export/home/user1/outputfile1
2. Click OK.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Continuous Monitoring Usage (see page 64)
JMS Publish and JMS Subscribe Jobs (see page 117)
Using a Namespace for a User that has Different Passwords (see page 63)

JMX Jobs

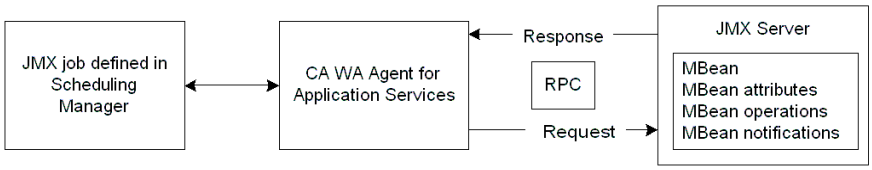
Java Management Extension (JMX) technology is included in the Java Standard Edition (SE) platform, version 5 and higher. JMX lets you remotely access applications, using a Remote Method Invocation (RMI) connector, for monitoring and management purposes.

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.

Define a JMX-MBean Attribute Get Job

You can define a JMX-MBean Attribute Get 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMX-MBean Attribute Get job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Attribute Get icon appears on the Application workspace view.

3. Right-click the JMX-MBean Attribute Get icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Attribute Get dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of an MBean. You can type the MBean name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the MBean. The format is the following:

`domain_name:key=value[,key=value...]`

domain_name

Specifies the default domain name.

***key=value*[*,key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

Attribute

Specifies the name of the MBean attribute you want to query or set. You can type the attribute name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the attribute.

Example: cachesize

5. (Optional) Specify the following additional information:

User

Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Success pattern

Specifies a Java regular expression to check against the job's output. If the output matches the success pattern, the job completes; otherwise, it fails.

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. Click OK.

   The JMX-MBean Attribute Get job is defined.

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.

To query a JMX server for the value of an MBean attribute

1. Enter the following required information in the Basic page:
   - Name—QUERY
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://localhost:9999/server
   - MBean—DefaultDomain:index=1,type=Config
   - Attribute—cachesize
2. Click OK.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Browse for an MBean (see page 153)
Browse for an MBean Attribute (see page 153)
JMX Jobs (see page 128)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a JMX-MBean Attribute Set Job

You can define a JMX-MBean Attribute Set 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMX-MBean Attribute Set job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Attribute Set icon appears on the Application workspace view.

3. Right-click the JMX-MBean Attribute Set icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Attribute Set dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of an MBean. You can type the MBean name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the MBean. The format is the following:

`domain_name:key=value[,key=value...]`

domain_name

Specifies the default domain name.

***key=value*[*,key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

Attribute

Specifies the name of the MBean attribute you want to query or set. You can type the attribute name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the attribute.

Example: cachesize

5. (Optional) Specify the following additional information:

User

Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Success pattern

Specifies a Java regular expression to check against the job's output. If the output matches the success pattern, the job completes; otherwise, it fails.

Note: Using the success pattern field adds a condition to the job. The job will only change the attribute to the new value if the current attribute value matches the success pattern you specify.

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. Click Edit to enter the parameter value you want to set.

   The Edit Parameter dialog opens.

7. Select one of the following parameter options:
   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

8. Complete the following fields that apply to the parameter option:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

9. Click OK.

   The Edit Parameter dialog closes and the parameter appears in a list.

10. Click OK.

    The JMX-MBean Attribute Set job is defined.

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 a value returned by a JMX-Attribute Set job named size.

To change the value of an MBean attribute

1. Enter the following required information in the Basic page:
   - Name—JMXSET
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
   - MBean—DefaultDomain:index=1,type=cdc.jmx.SimpleDynamic
   - Attribute—State
2. Click Edit, select the Payload producing job option button, and enter the following details:
   - Type—String
   - Payload producing job—size
3. Click OK twice.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Browse for an MBean (see page 153)
Browse for an MBean Attribute (see page 153)
JMX Jobs (see page 128)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a JMX-MBean Create Instance Job

You can define a JMX-MBean Create Instance 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMX-MBean Create Instance job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Create Instance icon appears on the Application workspace view.

3. Right-click the JMX-MBean Create Instance icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Create Instance dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of the MBean you want to create. The format is the following:

`domain_name:key=value[,key=value...]`

domain_name

Specifies the default domain name.

***key=value*[,*key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

Class name

Specifies the fully qualified Java class of the MBean object.

5. (Optional) Specify the following additional information:

User

Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

6. Click New to add a parameter value for the constructor of the Java class.

   The Add parameter dialog opens.

7. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

8. Complete the following fields that apply to the parameter option:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

   Array

   Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

   Example: 1, 2, 3

9. Click OK.

   The Add parameter dialog closes and the parameter appears in the Parameters table.

10. (Optional) Repeat Steps 6 through 9 to specify additional parameters.

11. Click OK.

    The JMX-MBean Create Instance job is defined.

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".

To create an MBean instance on a JMX server

1. Enter the following required information in the Basic page:
   - Name—CREATE
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
   - MBean—DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic
   - Class name—cdc.jmx.SimpleDynamic
2. Click New, select the Value option button, and enter the following details:
   - Type—java.lang.String
   - Value—Hello
3. Click OK twice.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
JMX Jobs (see page 128)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a JMX-MBean Operation Job

You can define a JMX-MBean Operation 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMX-MBean Operation job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Operation icon appears on the Application workspace view.

3. Right-click the JMX-MBean Operation icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Operation dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of an MBean. You can type the MBean name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the MBean. The format is the following:

`domain_name:key=value[,key=value...]`

domain_name

Specifies the default domain name.

***key=value*[*,key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

Operation

Specifies the MBean operation to be invoked. You can type the operation name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the operation.

Example: resetcache

5. (Optional) Specify the following additional information:

User

Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Success pattern

Specifies a Java regular expression to check against the job's output. If the output matches the success pattern, the job completes; otherwise, it fails.

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. Click New to enter the parameter value you want to set.

   The New Parameter dialog opens.

   Note: If you use the arrow to browse for the operation, the browse feature also updates the parameter types. You only need to enter the value for each type. If you do not browse for the operation, click New to add parameters.

7. Select one of the following parameter options:
   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

8. Complete the following fields that apply to the parameter option:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

9. Click OK.

   The New Parameter dialog closes and the parameter appears in a list.

10. Click OK.

    The JMX-MBean Operation job is defined.

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.

To invoke an operation on an MBean

1. Enter the following required information in the Basic page:
   - Name—RESET
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://localhost:9999/server
   - MBean—DefaultDomain:index=1,type=Config
   - Operation—resetmem
2. Click Edit, select the Payload producing job option button, and enter the following details:
   - Type—Integer
   - Value—50
3. Click OK twice.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Browse for an MBean (see page 153)
JMX Jobs (see page 128)
Browse for an MBean Operation (see page 145)
Using a Namespace for a User that has Different Passwords (see page 63)

Browse for an MBean Operation

Note: This procedure applies to JMX-MBean Operation jobs.

You can use the Browse MBean operations dialog to select an operation within the job definition.

Follow these steps:

1. Open the Basic page of the JMX job definition.
2. Enter the MBean name in the MBean field.
3. Click the arrow browse button next to the Operation field.

   The Browse MBean operations dialog opens.
4. Click Refresh.

   The JMX server returns a list of operations for the MBean.
5. Select the operation you want and click OK.

   The operation you selected appears in the Operation field on the Basic page.

Define a JMX-MBean Remove Instance Job

You can define a JMX-MBean Remove Instance 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.
2. Select the JMX-MBean Remove Instance job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Remove Instance icon appears on the Application workspace view.
3. Right-click the JMX-MBean Remove Instance icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Remove Instance dialog opens.
4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of an MBean. You can type the MBean name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the MBean. The format is the following:

`domain_name``:`*`key=value`*`[,`*`key=value`*`...]`

domain_name

Specifies the default domain name.

***key=value*[,*key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

5. (Optional) Specify the following additional information:

   User

   Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

6. Click OK.

   The JMX-MBean Remove Instance job is defined.

Example: Remove an MBean Instance from a JMX Server

Suppose that you want to remove an MBean instance.

To remove an MBean instance from a JMX server

1. Enter the following required information in the Basic page:
   - Name—REMOVE
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://agenttest:5099/jmxserver
   - MBean—DefaultDomain:index=CreateIns1,type=cdc.jmx.SimpleDynamic
2. Click OK.

More information:

Browse for an MBean (see page 153)
JMX Jobs (see page 128)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a JMX-MBean Subscribe Job

You can define a JMX-MBean Subscribe 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the JMX-MBean Subscribe job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The JMX-MBean Subscribe icon appears on the Application workspace view.

3. Right-click the JMX-MBean Subscribe icon, and select Edit from the pop-up menu.

   The Basic page of the JMX-MBean Subscribe dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote JMX server that advertises MBeans.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

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.

Example: service:jmx:rmi:///jndi/rmi://localhost:9999/server

MBean

Specifies the name of an MBean. You can type the MBean name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the MBean. The format is the following:

`domain_name:key=value[,key=value...]`

domain_name

Specifies the default domain name.

***key=value*[*,key=value*...]**

Specifies one or more key=value pairs.

Example: DefaultDomain:type=SimpleDynamic,index=3

5. Select one of the following filter types:
   - Attributes—Identifies one or more attributes to monitor for change.
   - Types—Identifies one or more types of notification to monitor.

   Note: If you do not specify a filter, the job will subscribe to all notifications for the MBean.
6. Complete the following field that applies to the filter type:

   Attribute

   Specifies the name of the MBean attribute you want to query or set. You can type the attribute name or, if you have a connection to the JMX server, click the arrow next to the field to browse for the attribute.

   Example: cachesize

Type

Specifies the type of notification to monitor. You can type the notification type and click Add or, if you have a connection to the JMX server, click Get to browse for the notification type. The browse feature returns a list of notification types from which you can select one or more types.

Example: jmx.attribute.change

7. (Optional) Specify the following additional information:

User

Specifies the JMX server user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Alert name

Specifies the Alert name to be executed each time the subscription is satisfied. Defines the job as continuous.

8. Click OK.

The JMX-MBean Subscribe job is defined.

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 changealert is sent.

1. Enter the following information in the Basic page:
   - Name—CHANGE
   - Agent name—APPAGENT
   - URL—service:jmx:rmi:///jndi/rmi://localhost:9999/server
   - MBean—DefaultDomain:index=1,type=Config
   - Alert name—changealert
2. Select the Attributes option button in the Filter section and enter **cachesize** in the Attribute field.
3. Click OK.

More information:

Browse for an MBean (see page 153)
JMX Jobs (see page 128)
Continuous Monitoring Usage (see page 64)
Browse for MBean Notifications (see page 152)
Browse for MBean Attributes (see page 151)
Using a Namespace for a User that has Different Passwords (see page 63)

Browse for MBean Attributes

Note: This procedure applies to JMX-MBean Subscribe jobs.

You can use the Browse MBean attributes dialog to select one or more MBean attributes within the job definition.

Follow these steps:

1. Ensure you are connected to the CA Workload Automation DE server.
2. Open the Basic page of the JMX-MBean Subscribe job definition.
3. Complete the required fields.
4. Select the Attributes option button in the Filter section and click Get.

   The Browse MBean attributes dialog opens.
5. Click Refresh.

   The JMX server returns a list of corresponding MBean attributes.
6. Select one or more MBean attributes and click OK.

   Note: You can hold the Shift key to select multiple consecutive attributes or hold the Ctrl key to select individual random attributes.

   The MBean attributes you selected appear in the Filter list on the Basic page.

Browse for MBean Notifications

Note: This procedure applies to JMX-MBean Subscribe jobs.

You can use the Browse MBean notifications dialog to select one or more MBean notifications within the job definition.

Follow these steps:

1. Ensure you are connected to the CA Workload Automation DE server.
2. Open the Basic page of the JMX-MBean Subscribe job definition.
3. Complete the required fields.
4. Select the Types option button in the Filter section and click Get.

   The Browse MBean notifications dialog opens.
5. Click Refresh.

   The JMX server returns a list of corresponding MBean notifications.
6. Select one or more MBean notifications and click OK.

   Note: You can hold the Shift key to select multiple consecutive notifications or hold the Ctrl key to select individual notifications.

   The MBean notifications you selected appear in the Filter list on the Basic page.

Browse for an MBean

Note: This procedure applies to all JMX job types except JMX-MBean Create Instance.

You can use the Browse MBeans dialog to locate an MBean within the job definition.

Follow these steps:

1. Ensure you are connected to the CA Workload Automation DE server.
2. Open the Basic page of the JMX job definition.
3. Click the arrow browse button next to the MBean field.

   The Browse MBeans dialog opens.
4. Enter the MBean name or a pattern to match MBean names against in the Name pattern field. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Note: You can specify a name pattern to match by clicking Get and selecting a pattern from the Domains field drop-down list.
5. Click Refresh.

   The JMX server returns a list of corresponding MBeans.
6. Select the MBean you want and click OK.

   The MBean you selected appears in the MBean field on the Basic page.

Browse for an MBean Attribute

Note: This procedure applies to the JMX-MBean Attribute Get and JMX-MBean Attribute Set jobs.

You can use the Browse MBean attributes dialog to select an attribute within the job definition.

Follow these steps:

1. Open the Basic page of the JMX job definition.
2. Enter the MBean name in the MBean field.
3. Click the arrow browse button next to the Attribute field.

   The Browse MBean attributes dialog opens.
4. Click Refresh.

   The JMX server returns a list of attributes for the MBean.
5. Select the attribute you want and click OK.

   The attribute you selected appears in the Attribute field on the Basic page.

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. For more information about installing the agent plug-in, see the *CA Workload Automation Agent for Application Services Implementation Guide*. For more information about configuring CA Workload Automation DE to work with Agents, see the *Implementation Guide*.
- 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*.

Define a POJO Job

You can define a POJO job to create a Java object instance with no arguments, invoke a method on the object instance, and store the method's output.

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

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the POJO job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The POJO icon appears on the Application workspace view.

3. Right-click the POJO icon, and select Edit from the pop-up menu.

   The Basic page of the POJO dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote server and invoke a method on a Java object.

   Class name

   Specifies the Java class to instantiate.

   Example: java.lang.String

   Method

   Specifies the Java method to call on the instance of the Java object.

   Example: concat

5. (Optional) Specify the following information:

   Output destination

   Specifies the location of the Java serialized object produced by the job.

   Limits: Up to 256 characters

   Note: If you do not specify an output destination, 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.

6. (Optional) Click Method Parameters in the left pane.

   The Method Parameters page opens in the right pane.

7. Click New to add parameters for the method.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value
   - Payload producing job—Identifies the name of a predecessor job
   - Array—Identifies more than one value

   The fields that are related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The POJO job is defined.

Example: Define a POJO Job

Suppose that you want to define a POJO job that creates an empty java string, and calls the method parseInt on it with the argument “5”.

Follow these steps:

1. Enter the following required information in the Basic page:
   - Name—IGNORE
   - Agent name—APPAGENT
   - Class name—java.lang.Integer
   - Method name—parseInt
2. Add the following parameter in the Method Parameters page:
   - Parameter Type—java.lang.String
   - Parameter Value—5
3. Click OK.

Example: Instantiate a New Class

Suppose that you want to instantiate a new class (Calculate) using a POJO job. In this example, the job calls the Sum method from the Calculate class. The Calculate class belongs to a package named Client.

Follow these steps:

1. Stop the agent.
2. Create the Classes directory in the following location:

   agent_install_dir\jars\ext
3. Create the Client directory in the following location:

   agent_install_dir\jars\ext\Classes

   Client is the name of the package that the Calculate class belongs to.
4. Copy the Calculate.class file into the Client directory that you created:

   agent_install_dir\jars\ext\Classes\Client
5. Start the agent.
6. Enter the following required information in the Basic page of the POJO job:
   - Name—TEST1
   - Agent name—APPAGENT
   - Class name—Client.Calculate
   - Method name—Sum
7. Add the following two parameters in the Method Parameters page:
   - Parameter Type—Int; Parameter Value—2
   - Parameter Type—Int; Parameter Value—3
8. Click OK.

Example: Instantiate a New Class Wrapped in a JAR File

Suppose that you want to instantiate a new class (Calculate) that is wrapped in a JAR file (axis.jar). In this example, the job calls the Sub method from the Calculate class. The Calculate class belongs to a package named Client.

Follow these steps:

1. Stop the agent.
2. Copy the axis.jar file in the following location:

   agent_install_dir\jars\ext

   The axis.jar file contains the Client package that the Calculate class belongs to.
3. Start the agent.
4. Enter the following required information in the Basic page of the POJO job:
   - Name—TEST2
   - Agent name—APPAGENT
   - Class name—Client.Calculate
   - Method name—Sub
5. Add the following two parameters in the Method Parameters page:
   - Parameter Type—Int; Parameter Value—5
   - Parameter Type—Int; Parameter Value—1
6. Click OK.

Example: Define a Payload Consuming Job

Suppose that you want to create a payload consuming job that uses a serialized Java object as an input parameter from a payload producing job.

In this example, the POJO job (POJO1) appends two string values by calling a method (append2Strings) from a class (QATest5) and returns a java.lang.String value to the Web Service RPC/Encoded job (WebService1). The class belongs to a package named Client. The WebService1 job performs Celsius to Fahrenheit conversion on the java.lang.String value that is returned from POJO1 job.

Note: Restart the agent after you deploy the QATest5 class file in the agent installation directory.

Follow these steps:

1. Open the application that you want to add the payload producing and payload consuming jobs to.
2. Enter the following required information in the Basic page of the POJO job:
   - Name—POJO1
   - Agent name—APPAGENT
   - Class name—Client.QATest5
   - Method name—append2Strings
3. Add the following two parameters in the Method Parameters page:
   - Parameter Type—String; Parameter Value—1
   - Parameter Type—String; Parameter Value—0
4. Click OK.
5. Enter the following required information in the Basic page of the Web Service RPC/Encoded job:
   - Name—WebService1
   - Agent name—WSAGENT
   - WSDL URL— http://vupan01-i72451/TempConvert.asmx?wsdl
   - Target namespace—http://tempuri.org/
   - Service name—TempConvert
   - Port name—TempConvertSoap
   - Operation—CelsiusToFahrenheit
   - End point URL—http://vupan01-i72451/TempConvert.asmx
6. Add the following parameter in the Operation Parameters page:
   - Payload producing job—POJO1

7. Add the following information in the Operation Response page:
   - Return class name—java.lang.String
   - Return xml type—String
   - Return namespace—http://tempuri.org/
8. Click OK.
9. Add the dependencies link from the POJO1 job to the WebService1 job.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
POJO Jobs (see page 154)

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.

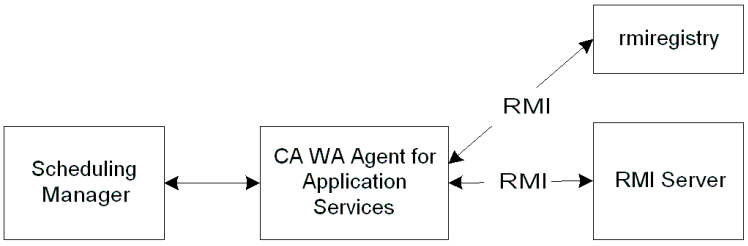
RMI jobs let you set up interaction between Java objects on different computers in a distributed network. Using an RMI job, you can access a remote server and invoke a method on a Java object. A method is a programmed procedure that is defined as a part of a Java class.

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:



Define an RMI Job

You can define the RMI 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the RMI job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The RMI icon appears on the Application workspace view.

3. Right-click the RMI icon, and select Edit from the pop-up menu.

   The Basic page of the RMI dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a remote server and invoke a method on a Java object.

RMI URL

Specifies the reference location of the object you want to invoke a method on. The format of an RMI URL is the following:

rmi://*host*:*port*/*name*

- (Optional) host is the name of the local or remote computer where the RMI registry that hosts the remote object runs. If you omit the host, the local host is used.
- (Optional) port is the RMI registry port number the host uses to listen for requests. If you omit the port, the default port of the host's RMI registry, 1099, is used.
- name is the name of a remote object.

Example: rmi://nissan:5000/test

Method name

Specifies the method of the remote Java class to invoke.

Example: update

5. (Optional) Specify the following additional information:

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. (Optional) Click Method Parameters in the left pane.

The Method Parameters page opens in the right pane.

7. Click New to add parameters for the method.

The New Parameter dialog opens.

8. Select one of the following parameter options:
   - Value—Identifies a single value
   - Payload producing job—Identifies the name of a predecessor job
   - Array—Identifies more than one value

The fields related to the option appear.

9. Complete the following fields that apply to the parameter option:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

   Array

   Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

   Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The RMI job is defined.

Example: Define an RMI Job

Suppose that you want to invoke a method that starts a remote server using remote object activation. You want the server to start immediately.

To define an RMI job

1. Enter the following required information in the Basic page:
   - Name—START
   - Agent name—AGENT
   - RMI URL—rmi://remotehost/Test
   - Method name—startserver
2. Add the following parameter in the Method Parameters page:
   - Type—String
   - Value—now
3. Click OK.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
RMI Jobs (see page 161)

Session Bean Jobs

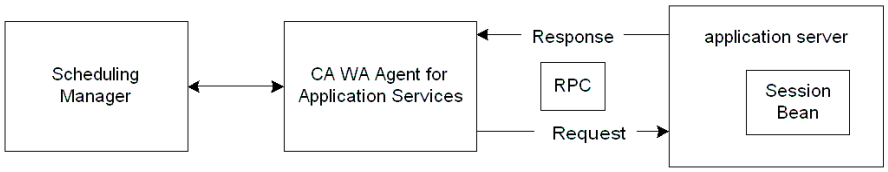
A session bean represents business logic or action to be taken (for example, charging a credit card or adding items to an online shopping cart).

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

Access a Stateless Session Bean Using a Session Bean Job

You can define a Session Bean job to access a stateless 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Session Bean job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Session Bean icon appears on the Application workspace view.

3. Right-click the Session Bean icon, and select Edit from the pop-up menu.

   The Basic page of the Session Bean dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a session bean on an application server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Bean name

Specifies the Session Bean Java Naming and Directory Interface (JNDI) name.

Example: test.HelloJndi

Method name

Specifies the method to be invoked on the application server.

Example: hello

5. (Optional) Specify the following additional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. (Optional) Click Method Parameters in the left pane.

   The Method Parameters page opens in the right pane.

7. Click New to define the parameters.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. Click OK.

    The Session Bean job is defined.

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, java.lang.String with a value a23. The output from the reverse method saves to the file C:\Makapt15. 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.

To invoke a method on a stateless session bean

1. Enter the following required information in the Basic page:
   - Name—REVERSE
   - Agent name—APPAGENT
   - Initial context factory—com.ibm.websphere.naming.WsnInitialContextFactory
   - Provider URL—iiop://172.24.0.0:2809
   - Bean name—CybEJBTestBean
   - Method name—reverse
2. Enter **C:\Makapt15** in the Output destination field.
3. Add the following parameter in the Method Parameters page:
   - Parameter Type—java.lang.String
   - Parameter Value—a23
4. Click OK.

   When the job runs, the output of the reverse method is stored in the output destination file.

More information:

Session Bean Jobs (see page 165)
Payload Producing and Payload Consuming Jobs (see page 71)
Using a Namespace for a User that has Different Passwords (see page 63)

Access a Stateful Session Bean Using a Session Bean Job

You can define a Session Bean job to access a stateful session bean, invoke a method on the bean, and return the results. For stateful session beans, you must define Create parameters. You can also define Method parameters if required.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Session Bean job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Session Bean icon appears on the Application workspace view.

3. Right-click the Session Bean icon, and select Edit from the pop-up menu.

   The Basic page of the Session Bean dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access a session bean on an application server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Bean name

Specifies the Session Bean Java Naming and Directory Interface (JNDI) name.

Example: test.HelloJndi

Method name

Specifies the method to be invoked on the application server.

Example: hello

Create method name

Defines the method name for a stateful session bean. Enter the method name preceded by the word create. This field is only required for stateful Session Bean jobs.

Example: createhello

5. (Optional) Specify the following additional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

6. Click Create Parameters in the left pane.

   The Create Parameters page opens in the right pane.

7. Click New to define the parameters.

   The New Parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

9. Complete the following fields, as required:

   Type

   Specifies the Java class of the parameter. This field is mandatory for the Value and Array parameter options only.

   Example: Integer

   Value

   (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

   Example: 2

   Payload producing job

   Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

   Note: The job must be defined within the same Application as the job you are defining.

   Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

10. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

11. (Optional) Click Method Parameters in the left pane.

    The Method Parameters page opens in the right pane.

12. Repeat Steps 7 through 10 to add method parameters.

13. Click OK.

    The Session Bean job is defined.

Example: Invoke a Method on a Stateful Session Bean

Suppose you want to access a stateful session bean for an online shopping cart. 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 chart with ISBN numbers 1551929120 and 1582701709.

To create a stateful Session Bean and invoke a method

1. Enter the following required information in the Basic page:
   - Agent name—APPAGENT
   - Initial context factory—com.ibm.websphere.naming.WsnInitialContextFactory
   - Provider URL—iiop://172.24.0.0:2809
   - Bean name—Shoppingcart
   - Create method name—Createaddbook
   - Method name—addbook
2. Add the following parameter in the Create Parameters page:
   - Parameter Type—String
   - Parameter Value—ISBN
3. Add the following array parameters in the Method Parameters page:
   - Parameter Type—Integer
   - Parameter Value—1551929120
   - Parameter Value—1582701709
4. Click OK.

   When the job runs, two books are added to the shopping cart.

More information:

Session Bean Jobs (see page 165)
Payload Producing and Payload Consuming Jobs (see page 71)
Using a Namespace for a User that has Different Passwords (see page 63)

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.

The following are the web service job types:

Web Service RPC/Encoded

Lets you call an operation within a web service and pass parameters to the operation using RPC/encoded style binding. The parameters can be actual values or a serialized Java object passed by another job.

Web Service Document/Literal

Lets you call an operation within a web service and pass parameters to the operation using document/literal style binding. The parameters represent a flattened view of the XML document the agent constructs. The values passed into the XML document can be literal 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*.

Configure the Client for a Proxy

If your company has a proxy set up to allow access to the Internet, you must configure the proxy in CA WA Desktop Client to use the wizard to define Web Service jobs. You require the proxy name and port for the configuration.

Follow these steps:

1. Select Window, Preferences.

   The Preferences dialog opens.

2. Click Workbench, General, Network Connections in the left pane.

   The Network Connections page opens in the right pane.

3. Select Manual proxy configuration, complete the proxy and port fields, and click OK.

   The proxy is configured in CA WA Desktop Client.

Note: CA WA Agent for Web Services must also be configured for the proxy.

More information:

Define an HTTP Job (see page 112)
Define a Web Service RPC/Encoded Job (see page 177)
Define a Web Service Document/Literal Job (see page 182)

Define a Web Service RPC/Encoded Job

You can define a Web Service RPC/Encoded job to call an operation within a web service and pass parameters to the operation using RPC/encoded style binding.

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

Follow these steps:

1. Configure CA WA Desktop Client for your company's proxy, if necessary, to access the Internet (see page 176).
2. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.
3. Select the Web Service job RPC/Encoded from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Web Service RPC/Encoded icon appears on the Application workspace view.
4. Right-click the Web Service RPC/Encoded icon, and select Edit from the pop-up menu.

   The Basic page of the Web Service RPC/Encoded dialog opens.
5. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access web services on a web server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Target namespace

   Specifies the target namespace used for the names of messages, port type, binding, and service defined in the web service WSDL. Complex data types such as arrays require the target namespace.

   Example: http://www.webserviceX.NET

Operation

Specifies the web service operation to be invoked.

Limits: Up to 256 characters

Example: GetQuote

6. (Optional) Specify the following additional information:

WSDL URL

Specifies the local path or remote URL for the WSDL file.

Clicking opens the Browse WSDL wizard, which lets you locate the WSDL file and select the required values.

Limits: Up to 2048 characters

Example: http://www.webservicex.com/stockquote.asmx?WSDL

Note: The easiest way to define a Web Service RPC/Encoded job is to use the wizard. Using the wizard, you can select the web service, port name, and operation for the WSDL document. You can also specify the parameter values for the operation. The WSDL-related fields are automatically populated in the job definition, including the fields in the Operation Parameters and Operation Response pages.

Service name

Specifies the web service name within the target namespace.

Limits: Up to 256 characters

Example: StockQuote

Port name

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 256 characters

Example: StockQuoteSoap

End point 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 2048 characters

Example: http://www.webservicex.net/stockquote.asmx

User

Specifies the user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Destination file

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

7. Click Operation Parameters in the left pane.

   The Operation Parameters page opens in the right pane.

8. Click New to define the parameters.

   The New Parameter dialog opens.

9. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.
   - Array—Identifies more than one value.

   The fields related to the option appear.

10. Complete the following fields, as required:

    Type

    Specifies the XML schema type of the parameter. This field is mandatory for the Value and Array parameter options only.

    Value

    (Optional) Specifies the String value for the method parameter. This field is used with the Value parameter option.

    Example: 2

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

Array

Specifies a set of string values for the method parameter. Click Add to enter each value. This field is used with the Array parameter option.

Example: 1, 2, 3

11. Click OK.

    The New Parameter dialog closes and the parameters appear in a list.

12. (Optional) Repeat Steps 8 through 11 to specify additional parameters.

13. (Optional) Click Operation Response in the left pane.

    The Operation Response page opens in the right pane.

    Note: Usually you do not require the information on this page if you have specified the WSDL URL.

14. Enter the following information:

    Return class name

    Specifies the Java class name of the return value.

    Return xml type

    Specifies the XML type that maps to returnclassname.

    Example: dateTime

    Return namespace

    Specifies the XML namespace for returnxmlname.

    Example: http://www.webserviceX.NET/

    Job criteria

    (Optional) Specifies the regular expression to determine if the return value is successful.

    Example: .+(CA)+.+

15. Click OK.

    The Web Service RPC/Encoded job is defined.

Example: Get New Year's Day for 2009

Suppose that you want to invoke a web service that returns the first day of the year for 2009. The URL for the WSDL that describes the web service and its location is http://www.holidaywebservice.com/Holidays/GBSCT/Dates/GBSCTHolidayDates.asmx?WSDL. The operation is named GetNewYear.

To get new year's day for 2009

1. Enter the following required information in the Basic page:
   - Name—DATE
   - Agent name—AGENT
   - Target namespace—http://www.27seconds.com/Holidays/GBSCT/Dates/
2. Enter **http://www.holidaywebservice.com/Holidays/GBSCT/Dates/GBSCTHolidayDates.asmx?WSDL** in the WSDL URL field and click the Browse button at the end of the field.
3. Select the following values in the Browse WSDL dialog:
   - Service name—GBSCTHolidayDates
   - Port name—GBSCTHolidayDatesSoap
   - Operation—GetNewYear
4. Click Edit to enter **2009** as the value of the parameter.
5. Click Finish to exit the Browse WSDL dialog.

When the job completes, a serialized object, java.util.GregorianCalendar, is sent to the job spool directory.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Web Service Jobs (see page 175)

Define a Web Service Document/Literal Job

You can define a Web Service Document/Literal job to call an operation within a web service and pass parameters to the operation using document/literal style binding.

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

Follow these steps:

1. Configure CA WA Desktop Client for your company's proxy, if necessary, to access the Internet (see page 176).
2. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.
3. Select the Web Service Document/Literal job from the Application and Web Services group in the Palette view, and drag the job to the workspace.

   The Web Service Document/Literal icon appears on the Application workspace view.
4. Right-click the Web Service Document/Literal icon, and select Edit from the pop-up menu.

   The Basic page of the Web Service Document/Literal dialog opens.
5. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that lets you access web services on a web server.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

WSDL URL

Specifies the local path or remote URL for the WSDL file.

Clicking opens the Browse WSDL wizard, which lets you locate the WSDL file and select the required values.

Limits: Up to 2048 characters

Example: http://www.webservicex.com/stockquote.asmx?WSDL

Note: The easiest way to define a Web Service Document/Literal job is to use the wizard. Using the wizard, you can select the web service, port name, and operation for the WSDL document. The WSDL-related fields are automatically populated in the job definition.

Service name

Specifies the web service name within the target namespace.

Limits: Up to 256 characters

Example: StockQuote

Port name

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 256 characters

Example: StockQuoteSoap

Operation

Specifies the web service operation to be invoked.

Limits: Up to 256 characters

Example: GetQuote

End point 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 2048 characters

Example: http://www.webservicex.net/stockquote.asmx

6. (Optional) Specify the following additional information:

User

Specifies the user under which the job will execute. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Destination file

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

7. Click Operation Parameters in the left pane.

The Operation Parameters page opens in the right pane.

8. Click New to define the parameters.

The Add parameter dialog opens.

9. Select one of the following parameter options:

- Value—Identifies a single value.
- Payload producing job—Identifies the name of a predecessor job.
- Array—Identifies more than one value.

The fields related to the option appear.

10. Complete the following fields, as required:

Name

Specifies the name of path of the element.

Value

(Optional) Specifies the scalar string value.

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

Array

Specifies an array of scalar string values. Click Add to enter each value. This field is used with the Array parameter option.

Notes:

- To specify an XML hierarchy of the document, specify multiple parameters.
- In the XML document, these parameters are represented in a flattened view. Consider the following example:

```
<tag1>
    <tag2>value</tag2>
<tag1>
```

In this example, the full path to the value is /tag1/tag2.

- For tags that do not hold any value, the full path is required because the no-value tags could represent arrays of structures and appear in the multiple levels.
- For parameters that contain attributes, the attributes are added to the full path of the parameter separating them with the '@' symbol. Consider the following example:

```
<tag1>
    <tag2 key="thetag">value</tag2>
</tag1>
```

In this example, the list of parameters would be as follows:

  - Parameter Name—/tag1; Parameter Value—(No value; leave empty)
  - Parameter Name—/tag1/tag2; Parameter Value—value
  - Parameter Name—/tag1/tag2@key; Parameter Value—thetag

11. Click OK.

    The Add parameter dialog closes and the parameters appear in a list.

12. (Optional) Repeat Steps 8 through 11 to specify additional parameters.

13. Click Operation Response in the left pane.

    The Operation Response page opens in the right pane.

14. (Optional) Complete the following Job criteria fields to define success criteria for the job:

    Filter

    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.

    Example: .*42.*

    Limits: Up to 512 characters

    Note: 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.

    XPath

    (Optional) Checks the xpath expression that is used to evaluate a return string.

    Example: //ns:return

    Limits: Up to 512 characters

    Notes:

    - If the xpath expression is specified, the filter expression is required. If the xpath expression is not specified, the filter expression applies to the entire response returned by the call.
    - The response is transformed into a formatted XML document before being checked against the filter. The agent formats the returned XML document to be easier to read with a four-space indent for nested tags. Specifically, the following keys are used for transformation:
      - “indent” : "yes"
      - “standalone” : “yes”
      - "{http://xml.apache.org/xslt}indent-amount" : "4"
    - For Web Service Document/Literal payload producing jobs, only the portion of the returned data that satisfies the success criteria is stored in the output file for use as payload.

15. Click Connection Information in the left pane if connection authentication is required.

    The Connection Information page opens in the right pane.

16. Complete the following fields that are required by a website with connection authentication. Speak to your agent administrator for details.

    Client authentication user

    Specifies the user name required by a website for connection authentication. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

    Limits: Up to 128 characters

    Examples: Bob, Production:Bob

    Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

    Origin host name

    Specifies the origin host name for NTLM connection authentication. If unspecified, the origin defaults to the computer name where the agent is running.

    Limits: Up to 128 characters

    NTLM authentication domain

    Specifies the domain for NTLM connection authentication.

    Limits: Up to 128 characters

    Authentication order

    Specifies a list of protocols that is used by a web server for authentication.

    Valid values are as follows:

    - BASIC
    - DIGEST
    - NTLM

    Notes:

    - You can specify any of the following protocols in any order: BASIC, DIGEST, and NTLM. To specify more than one protocol, separate the protocols by commas.
    - If you are connecting to an old web server that cannot negotiate authentication protocols, enter the following list in the specified order: BASIC, DIGEST, NTLM.

17. Click Proxy Information in the left pane if proxy authentication is required.

    The Proxy Information page opens in the right pane.

18. Complete the following fields that are required for proxy authentication. Speak to your agent administrator for details.

    Don't use global proxy defaults

    Ignores the global proxy configuration specified by the proxy parameters in the agentparm.txt file. Select this option if you want to ignore the proxy for this job only. For example, you might ignore the proxy if the request is going to a server on the LAN that does not require the proxy.

    Proxy host

    Specifies the proxy host name to use for the request.

    Limits: Up to 128 characters; case-sensitive

    Note: Your agent administrator can specify a default setting for all Web Service Document/Literal jobs by setting the http.proxyHost parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

    Proxy port

    Specifies the proxy port to use for the request.

    Limits: Up to 16 digits

    Note: Your agent administrator can specify a default setting for all Web Service Document/Literal jobs by setting the http.proxyPort parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

    Proxy user

    Specifies the user name required for proxy authentication. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

    Limits: Up to 128 characters; case-sensitive; it cannot contain delimiters (such as spaces)

    Examples: Bob, Production:Bob

    Notes:

    - The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
    - Your agent administrator can specify a default setting for all Web Service Document/Literal jobs by setting the http.proxyUser parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

Proxy domain

Specifies the domain for proxy authentication. This field is used for NTLM authentication.

Limits: Up to 128 characters; case-sensitive

Note: Your agent administrator can specify a default setting for all Web Service Document/Literal jobs by setting the http.proxyDomain parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

Origin host name (proxy)

Specifies the origin host name for proxy authentication. This field is used for NTLM authentication.

Limits: Up to 128 characters; case-sensitive

Note: Your agent administrator can specify a default setting for all Web Service Document/Literal jobs by setting the http.proxyOrigin parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

19. Click OK.

The Web Service Document/Literal job is defined.

Example: Invoke a Web Service to Get a Stock Quote

The following job calls the GetPrice operation in the StockQuoteService service on the server at http://wsservice:8080/axis2/services/StockQuoteService, using the default proxy settings from the agentparm.txt file.

The job passes in the following XML:

```
<getPrice><symbol>ABC</symbol></getPrice>
```

The job checks the returned XML for the value in an element that matches the criteria xpath //ns:return. If the return values contain 42, the job completes successfully. Otherwise, the job fails.

To invoke a web service to get a stock quote

1. Enter the following required information in the Basic page:
   - Name—sq1
   - Agent name—WSAGENT
   - WSDL URL—http://www.webservicex.net/stockquote.asmx?WSDL
   - Service name—StockQuoteService
   - Port name—StockQuoteServiceHttpSoap11Endpoint
   - Operation—GetPrice
   - End point URL—http://wsservice:8080/axis2/services/StockQuoteService
2. Add the following parameters in the Operation Parameters page:
   - Web service parameter 1:
     - Parameter Name—/getPrice
     - Defined as—Value
     - Parameter Value—(No value; leave empty)
   - Web service parameter 2:
     - Parameter Name—/getPrice/symbol
     - Defined as—Value
     - Parameter Value—ABC
3. Enter the following information in the Operation Response page:
   - Filter—.*42.*
   - XPath—//ns:return
4. Click OK.

   When the job completes, the spool file will contain the stock quote.

Example: Add Two Numbers Passing Two Parameters

This job calls a simple calculator web service at http://ws1.parasoft.com/glue/calculator. The web service requests the "add" operation to add two numbers, passing in two parameters: "x", which is the result from the last execution of the getmax1 job and "y", which is the literal value 1.00. The job uses specific values for the HTTP proxy host, HTTP proxy port, proxy authentication user, and proxy authentication domain.

To add two numbers passing two parameters

1. Enter the following required information in the Basic page:
   - Name—incpayload
   - Agent name—WSAGENT
   - WSDL URL—http://soaptest.parasoft.com/calculator.wsdl
   - Service name—Calculator
   - Port name—ICalculator
   - Operation—add
   - End point URL—http://ws1.parasoft.com/glue/calculator
2. Add the following parameters in the Operation Parameters page:
   - Web service parameter 1:
     - Parameter Name—/add
     - Defined as—Value
     - Parameter Value—(No value; leave empty)
   - Web service parameter 2:
     - Parameter Name—/add/x
     - Defined as—Payload producing job
     - Job—getmax1

- Web service parameter 3:
  - Parameter Name—/add/y
  - Defined as—Value
  - Parameter Value—1.00

3. Select the Don't use global proxy defaults check box in the Proxy Information page.
4. Enter the following information in the Proxy Information page:
   - Proxy host—141.202.248.209
   - Proxy port—80
   - Proxy user—myuser@mydomain
   - Proxy domain—mydomain
5. Click OK.

   When the job completes, the spool file will contain the result of the calculation.

Example: Invoke a Web Service to Get Movie Information

This job calls the GetTheatersAndMovies operation in the MovieInformation service hosted at http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx, requesting information on movies showing around the 12590 zip code. It passes the request through the proxyhost.example.com proxy on port 80, and for proxy authentication specifies the user myuser@mydomain, the domain mydomain, and the origin host name of myhost.example.com.

To invoke a web service to get movie information

1. Enter the following required information in the Basic page:
   - Name—showing
   - Agent name—WSAGENT
   - WSDL URL—http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx?wsdl
   - Service name—MovieInformation
   - Port name—MovieInformationSoap
   - Operation—GetTheatersAndMovies
   - End point URL—http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx

2. Add the following parameters in the Operation Parameters page:
   - Web service parameter 1:
     - Parameter Name—/GetTheatersAndMovies
     - Defined as—Value
     - Parameter Value—(No value; leave empty)
   - Web service parameter 2:
     - Parameter Name—/GetTheatersAndMovies/zipCode
     - Defined as—Value
     - Parameter Value—12590
   - Web service parameter 3:
     - Parameter Name—/GetTheatersAndMovies/radius
     - Defined as—Value
     - Parameter Value—20
3. Enter the following information in the Proxy Information page:
   - Proxy host—proxyhost.example.com
   - Proxy port—80
   - Proxy user—myuser@mydomain
   - Proxy domain—mydomain
   - Origin host name (proxy)—myhost.example.com
4. Click OK.

   When the job completes, the spool file will contain the movie information.

Example: Invoke a CA Workload Automation DE Web Service over HTTPS

This job invokes a CA Workload Automation DE web service to create a global variable in the DEFAULT context. It assumes that CA WA Web Services is installed on a host named workstation and is listening on the 8443 port.

Since HTTPS is used to communicate with the web service, the following truststore parameters must be set in the agentparm.txt file on the agent:

https.client.ssl.truststore

Specifies the full path to the cacerts file on the agent, including the filename cacerts.

Example: C:\Program Files\CA\Workload Automation R11.3 SP1_b5\ESPSystemAgent\cacerts

Note: Import the SSL certificate into the cacerts file on the agent using the keytool utility that is provided with the JRE.

https.client.ssl.truststore.password

Specifies the password for accessing the truststore file. The password is encrypted in hexadecimal form.

Example: E371BBD062764757

Note: The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the *CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide*.

To invoke a CA Workload Automation DE web service over HTTPS

1. Enter the following required information in the Basic page:
   - Name—createVariable
   - Agent name—WSAGENT
   - WSDL URL—https://workstation:8443/axis2/services/EspDSeriesService?wsdl
   - Service name—EspDSeriesService
   - Port name—EspDSeriesServiceHttpSoap11Endpoint
   - Operation—createVariable
   - End point URL—https://workstation:8443/axis2/services/EspDSeriesService
2. Add the following parameters in the Operation Parameters page:
   - Web service parameter 1:
     - Parameter Name—/createVariable
     - Defined as—Value
     - Parameter Value—(No value; leave empty)

- Web service parameter 2:
  - Parameter Name—createVariable/name
  - Defined as—Value
  - Parameter Value—level
- Web service parameter 3:
  - Parameter Name—/createVariable/value
  - Defined as—Value
  - Parameter Value—2
- Web service parameter 4:
  - Parameter Name—/createVariable/context
  - Defined as—Value
  - Parameter Value—DEFAULT

3. Enter the following information in the Connection Information page:
   - Client authentication user—schedmaster

     Note: The credentials of this user must be defined in the Topology under the agent that you specified in the job definition.
   - Authentication order—BASIC
4. Click OK.

   When the job runs, a global variable named level is created in the DEFAULT context with a value of 2.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Web Service Jobs (see page 175)

Specify WS-Security Options in a Web Service Document/Literal Job

For Web Service Document/Literal jobs, you can specify the following additional SOAP-level security for the web services messages:

- Transport-level authentication
  - Usertoken—a user name token and password are added to each message and validated in the web service's SOAP layer.
  - Timestamp—a timestamp, with a time-to-live parameter, is added to each message to prevent message hijacking and reuse.
- Message-level security
  - Signature—messages are signed with a certificate.
  - Encryption—messages are encrypted at the message level, increasing security when going through proxies that may require decrypting or re-encrypting of the transport stream.

Note: For more information about controlling the agent's use of certificates for the Signature and Encryption WS-Security options, see the *CA Workload Automation Agent for Web Services Implementation Guide*.

Follow these steps:

1. Open the Web Service Document/Literal job that you want to specify WS-Security options for.

   The Basic page of the Web Service Document/Literal dialog opens.

2. Click WS-Security options in the left pane.

   The WS-Security opens page opens in the right pane.

3. Click New to define the WS-Security options.

   The Add parameter dialog opens.

4. Select one of the following WS-Security options and then complete the applicable fields:

   Usertoken

   Adds a user name and password to the SOAP message that is validated in the server's SOAP layer.

   Timestamp

   Adds a timestamp to the SOAP message with an optional time-to-live option to prevent reuse of old messages.

Signature

Adds a digital signature to the SOAP message, using the certificate specified in the agentparm.txt file.

Note: Your agent administrator must specify the ws.wss.keystore, ws.wss.keystore.alias, and ws.wss.keystore.password parameters in the agentparm.txt file.

Encryption

Encrypts the message. The server will decrypt the message in the SOAP layer.

Note: Your agent administrator must specify the ws.wss.keystore, ws.wss.keystore.alias, and ws.wss.keystore.password parameters in the agentparm.txt file if wss_keytype is ISSUER_SERIAL.

5. Complete the following fields, depending on the WS-Security option you selected. The Signature option does not have any applicable fields.

The following fields apply to the Usertoken option:

User

Specifies the user name that is added to the SOAP message. The user must be defined in the Topology.

Digest

Specifies whether you want to pass the password in digest form. Valid values are as follows:

true

Sends the password in the encrypted form to the SOAP request.

false

Send the password in the plain form to the SOAP request.

Default: true

Important! We recommend that you set Digest to true. If Digest is set to false, the agent writes the decrypted plain text password into the spool file.

The following fields apply to the Timestamp option:

Timestamp

Specifies the time-to-live in seconds.

The following fields apply to the Encryption option:

Service Alias

Specifies the alias in the keystore defined by properties for the Web Service certificate.

Limits: Up to 512 characters

Select one of the following:

Issuer Serial

Uses X509 certificate for encryption. It is obtained from the keystore defined by the properties specified in the agentparm.txt file.

Embedded Keyname

Uses the following key specified in the job definition for encryption:

Keyname

Specifies the key name as defined by the web service to be invoked.

Keyvalue

Specifies the key in hexadecimal format as defined by the web service to be invoked. The agent sends the key encrypted.

Limits: Up to 2048 characters

6. Click OK to save the security parameter.
7. (Optional) Repeat Steps 3-6 to specify additional WS-Security options.
8. Click OK.

The WS-Security options are specified.

Example: Get Maximum Number

This job calls the get maximum number web service at http://thismach:8080/axis2/services/TestWS. The web service requests the "intGetMax" operation to get the maximum number, passing in three parameters. The Usertoken and Timestamp WS-Security options specify transport level authentication. The Signature and Encryption WS-Security options specify message level security.

To get maximum number

1. Enter the following required information in the Basic page:
   - Name—getmax2
   - Agent name—WSAGENT
   - WSDL URL—http://thismach:8080/axis2/services/TestWS?wsdl
   - Service name—TestWS
   - Port name—TestWSHttpSoap11Endpoint
   - Operation—intGetMax
   - End point URL—http://thismach:8080/axis2/services/TestWS
2. Enter **causer** in the User field.
3. Add the following parameters in the Operation Parameters page:
   - Web service parameter 1:
     - Parameter Name—/intGetMa
     - Defined as—Value
     - Parameter Value—(No value; leave empty)
   - Web service parameter 2:
     - Parameter Name—/intGetMax/ii[3]
     - Defined as—Value
     - Parameter Value—"1,4,2"
4. Add the following parameters in the WS-Security options page:
   - Security parameter 1:
     - Defined as—Usertoken
     - Values—User="myuser@hostname",Digest="TRUE"

- Security parameter 2:
  - Defined as—Timestamp
  - Values—TTL="120"
- Security parameter 3:
  - Defined as—Signature
- Security parameter 4:
  - Defined as—Encryption
  - Values—ServiceAlias="securestore",KeyType="EMBEDDED_KEYNAME",KeyName=SessionKey",KeyValue="31FDCBDAFBCD6BA8E619A7BF51F7C73E"

5. Enter the following information in the Operation Response page:
   - Filter—.*42.*
   - XPath—//ns:return
6. Enter the following information in the Connection Information page:
   - Authentication order—DIGEST,BASIC,NTLM
7. Click OK.

   When the job completes, the spool file will contain the maximum number.

CA WA Jobs

CA WA jobs let you define Links and Tasks in your Applications. A Link is a workload object that completes immediately after its predecessor dependencies are met. A Task is a workload object that requires manual completion.

Create a Link

Links can have many uses in your Applications. For example, you can use links to simplify interlinked dependencies, schedule a cyclic Application, or keep an Application active. The server completes a link as soon as its dependencies are met.

Note: For more information on using links, see the *Examples Cookbook*.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Link workload object from the CA WA group in the Palette view, and drag the workload object to the workspace.

   The Link icon appears on the Application workspace view.

3. Right-click the Link icon, and select Edit from the pop-up menu.

   The Basic page of the Link dialog opens.

4. Enter the name of the link, specify a run frequency and other parameters as appropriate, and click OK.

   The link is defined.

More information:

Interlinked Dependencies (see page 550)

Create a Task

Tasks can have many uses in your Applications. For example, you can use tasks to represent a manual dependency, run different jobs based on the time a predecessor job completes, or delay job submission until the next hour. You must complete a task for its successors to become eligible for submission.

Note: For more information on using tasks, see the *Examples Cookbook*.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Task workload object from the CA WA group in the Palette view, and drag the workload object to the workspace.

   The Task icon appears on the Application workspace view.

3. Right-click the Task icon, and select Edit from the pop-up menu.

   The Basic page of the Task dialog opens.

4. Enter the name of the task, specify a run frequency and other parameters as appropriate, and click OK.

   The task is defined.

More information:

Manual Dependencies (see page 551)

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:

DB Stored Procedure

Lets you run a stored procedure.

DB Trigger

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

DB Monitor

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

SQL

Lets you execute an SQL statement.

Define a DB Stored Procedure Job

You can define a DB Stored Procedure 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.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the DB Stored Procedure job from the Database group in the Palette view, and drag the job to the workspace.

   The DB Stored Procedure icon appears on the Application workspace view.

3. Right-click the DB Stored Procedure icon, and select Edit from the pop-up menu.

   The Basic page of the DB Stored Procedure dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Database Agent that runs the job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Procedure name

Specifies the name of the procedure to run.

5. (Optional) Specify the following additional information:

DB user

Specifies the database user the job runs under. The database user specified in the job definition overrides the default specified in the agentparm.txt file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Examples: Bob, Production:Bob

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges. For example, if a job requires sysdba privileges, you would enter **as sysdba** in this field.

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@`*`host`*`:`*`port`*`:`*`dbname`*

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://`*`host`*`:`*`port`*`;DatabaseName=`*`dbname`*

For an IBM DB2 database, use the following format:

`jdbc:db2://`*`host`*`:`*`port`*`/`*`dbname`*

Return data type

Specifies the variable type to be returned.

Success criteria

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.

Notes:

- This field only applies to SQL queries that are SELECT statements.
- 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.
- 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.
- Some characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\). For example, to match the characters *.* literally, specify *\.* in your regular expression. The backward slashes escape the characters' special meanings.

6. (Optional) Click Parameters in the left pane.

   The Parameters page opens in the right pane.

7. Click Add to send parameters to the procedure, and complete the following fields:

   Parameter

   Specifies the parameter variable that is passed to the procedure.

   Note: You must specify the parameter name, whether the parameter is an input parameter (in), an output parameter (out) or both (in out), and the parameter type.

   Example: onea in out CHAR

   Value

   Defines the parameter value passed to the procedure.

8. Click OK.

   The DB Stored Procedure job is defined.

Example: Run a Stored Procedure

Suppose that you want a job named DBSP to invoke the PAYROLL procedure and check whether the employee name (ename) starts with John. The job uses the default values for the DB user and the DB URL fields. The parameter ename, which is both an input and output parameter of type VARCHAR, is passed to the procedure when it runs with value John Evans. This job runs on the DBAGENT agent computer.

To run a stored procedure

1. Enter the following information in the Basic page:
   - Name—DBSP
   - Agent name—DBAGENT
   - Procedure name—PAYROLL
   - Success criteria—ename=John.*
2. Add the following parameter in the Parameters page:
   - Parameter—ename in out VARCHAR
   - Value—John Evans
3. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

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

JDBC Data Type	Valid Input Format	Oracle	MS SQL Server	DB2
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
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;
```

DB Monitor and DB Trigger Jobs

You can monitor database changes either by using a DB Trigger job or a DB Monitor job.

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

The DB Trigger job offers the following advantages over DB Monitor jobs:

- You can monitor for more conditions. With DB Trigger jobs, you can monitor for rows added, deleted, and updated. With DB Monitor jobs, you can only monitor for rows added and deleted.
- DB Trigger jobs detect all changes made to the database; DB Monitor jobs monitor for changes only in 10 second intervals, by default.

Suppose that you want to send an Alert each time a new row is added. Within a 10 second interval, assume a row is added while another row is deleted. A DB Trigger job sends the Alert when the new row is detected, but the DB Monitor job may not send the Alert because no change in the total number of rows was detected.

Each DB Trigger job, however, creates a database trigger on the database. You should contact your database administrator before using a DB Trigger job.

Note: A table being monitored should not be dropped, because the DB Trigger or DB Monitor job remains in EXEC state even if the table has been dropped.

Define a DB Trigger Job

You can define a DB Trigger 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the DB Trigger job from the Database group in the Palette view, and drag the job to the workspace.

   The DB Trigger icon appears on the Application workspace view.

3. Right-click the DB Trigger icon, and select Edit from the pop-up menu.

   The Basic page of the DB Trigger dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Database Agent that monitors database table changes.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Table name

   Specifies the name of the table to be monitored in the database.

Trigger type

Specifies the type of change monitored (Insert, Delete, or Update).

Note: You can specify multiple trigger types for Oracle and Microsoft SQL Server. For Oracle, separate multiple types with "or", for example, **Insert or Delete**. For Microsoft SQL Server, separate multiple types with a comma, for example, **Insert, Delete**.

5. (Optional) Specify the following additional information:

DB user

Specifies the database user the job runs under. The database user specified in the job definition overrides the default specified in the agentparm.txt file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Examples: Bob, Production:Bob

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges. For example, if a job requires sysdba privileges, you would enter **as sysdba** in this field.

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@host:port:dbname`

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://host:port;DatabaseName=dbname`

For an IBM DB2 database, use the following format:

`jdbc:db2://host:port/dbname`

Trigger condition

Specifies the condition to monitor within the database. For Oracle and DB2, this is the WHEN clause. For Microsoft SQL Server, this is the IF clause. For the specific database syntax, refer to your database vendor's documentation.

Note: You can use a JavaScript symbolic variable in this field, preceded by the percent sign (%), for example, %TEST. In SQL, the percent sign represents a wildcard character. To use a wildcard character in this field, escape the percent sign by doubling it, for example, %%TEST.

Alert

Specifies the name of the Alert used for continuous monitoring. Each time the monitor or trigger condition is satisfied, the server triggers the Alert.

6. Click OK.

   The DB Trigger job is defined.

Example: Monitor a Database Table for Updated Rows with a Trigger Condition

Suppose that you want a job named DBDT to monitor the table Products for updates on an MS SQL Server database. If the number of units of product 65 falls below 50, the server triggers the predefined Alert valueLow. The user sa owns the Products table and is authorized to create triggers on the database. This job runs on the DBAGENT agent computer.

To monitor a database table for updated rows with a trigger condition

1. Enter the following information in the Basic page:
   - Name—DBDT
   - Agent name—DBAGENT
   - DB user—sa
   - DB URL—jdbc:sqlserver://10.1.1.10:1433;DatabaseName=Northwind
   - Table name—Products
   - Trigger type—Update
   - Trigger condition—(select UnitsInStock from Products where ProductID = 65)<50
   - Alert—valueLow
2. Click OK.

More information:

Continuous Monitoring Usage (see page 64)
DB Monitor and DB Trigger Jobs (see page 209)
Using a Namespace for a User that has Different Passwords (see page 63)

Define a DB Monitor Job

You can define a DB Monitor 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the DB Monitor job from the Database group in the Palette view, and drag the job to the workspace.

   The DB Monitor icon appears on the Application workspace view.

3. Right-click the DB Monitor icon, and select Edit from the pop-up menu.

   The Basic page of the DB Monitor dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Database Agent that monitors database additions and deletions.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Table name

   Specifies the name of the table to be monitored in the database.

Monitor type

Specifies the type of database change to monitor (INCREASE, DECREASE, INCREASE or DECREASE).

5. (Optional) Specify the following additional information:

DB user

Specifies the database user the job runs under. The database user specified in the job definition overrides the default specified in the agentparm.txt file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Examples: Bob, Production:Bob

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges. For example, if a job requires sysdba privileges, you would enter **as sysdba** in this field.

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@host:port:dbname`

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://host:port;DatabaseName=dbname`

For an IBM DB2 database, use the following format:

`jdbc:db2://host:port/dbname`

Monitor condition

Specifies the condition to monitor within the database. This is equivalent to an SQL where clause.

Note: You can use a JavaScript symbolic variable in this field, preceded by the percent sign (%), for example, %TEST. In SQL, the percent sign represents a wildcard character. To use a wildcard character in this field, escape the percent sign by doubling it, for example, %%TEST.

Alert

Specifies the name of the Alert used for continuous monitoring. Each time the monitor or trigger condition is satisfied, the server triggers the Alert.

6. Click OK.

   The DB Monitor job is defined.

Example: Monitor Database Changes Using a Database Monitor Job

Suppose that you want a job named DBDM to monitor the table HOURLYWAGE for an increase in the number of rows. AdminUser is not the database user provided during installation. If the hourly wage of an employee is greater than 30, the server triggers the predefined Alert valueAboveNorm. This job runs on the DBAGENT agent computer.

To monitor database changes using a Database Monitor job

1. Enter the following information in the Basic page:
   - Name—DBDM
   - Agent name—DBAGENT
   - DB user—AdminUser
   - Table name—HOURLYWAGE
   - Monitor type—INCREASE
   - Monitor condition—wage>30
   - Alert—valueAboveNorm
2. Click OK.

More information:

Continuous Monitoring Usage (see page 64)
DB Monitor and DB Trigger Jobs (see page 209)
Using a Namespace for a User that has Different Passwords (see page 63)

Define an SQL Job

You can define an SQL 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective..

   The Application appears in the workspace.

2. Select the SQL job from the Database group in the Palette view, and drag the job to the workspace.

   The SQL icon appears on the Application workspace view.

3. Right-click the SQL icon, and select Edit from the pop-up menu.

   The Basic page of the SQL dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Database Agent that runs the job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

SQL command

Specifies the SQL statement to be run.

Note: You can use a JavaScript symbolic variable in this field, preceded by the percent sign (%), for example, %TEST. In SQL, the percent sign represents a wildcard character. To use a wildcard character in this field, escape the percent sign by doubling it, for example, %%TEST.

5. (Optional) Specify the following additional information:

DB user

Specifies the database user the job runs under. The database user specified in the job definition overrides the default specified in the agentparm.txt file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Examples: Bob, Production:Bob

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges. For example, if a job requires sysdba privileges, you would enter **as sysdba** in this field.

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@`*`host`*`:`*`port`*`:`*`dbname`*

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://`*`host`*`:`*`port`*`;DatabaseName=`*`dbname`*

For an IBM DB2 database, use the following format:

`jdbc:db2://`*`host`*`:`*`port`*`/`*`dbname`*

Output file

Defines the output file to store the results of the SQL query.

Default: Spool file

Success criteria

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.

Notes:

- This field only applies to SQL queries that are SELECT statements.
- 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.
- 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.
- Some characters have a special meaning in regular expressions. To use these characters literally, precede the characters with one backward slash (\). For example, to match the characters *.* literally, specify *\.* in your regular expression. The backward slashes escape the characters' special meanings.

6. Click OK.

   The SQL job is defined.

Example: Run an SQL Query against an Oracle Database

Suppose that you want a job named SQL to run a query against an Oracle database and store the results in the job's spool file on the DBAGENT agent computer.

To run an SQL query against an Oracle database

1. Enter the following information in the Basic page:
   - Name—SQL
   - Agent name—DBAGENT
   - DB URL—jdbc:oracle:thin:@10.1.1.10:1521:ESP
   - SQL command—select jobname from tbtco
2. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

External Jobs

External jobs let you create dependencies between two jobs in different Applications. You can create dependencies between Applications running on the same scheduling manager or different scheduling managers.

Define an External Job

You can define an external job to create a dependency between two jobs in different Applications.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the External-Other Scheduler or External-Same Scheduler object from the External group in the Palette view, and drag the job to the workspace.

   Note: If the external job runs on the same scheduling manager as the home Application, use the External-Same Scheduler job. If the external job runs on a different scheduling manager than the home Application, use the External-Other Scheduler job.

   The external icon appears on the Application workspace view.

3. Right-click the external icon, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

4. Complete the following fields:

   Name

   Specifies the name of the external job. It must be the same name as the job in the home Application.

   Qualifier

   (Optional) Specifies the qualifier of the external job. It must be the same qualifier as the job in the home Application.

   Home Application name

   (Optional) Specifies the name of the home Application that submits the job. The server submits the external job in the distant Application if the Application you specify submits the job.

At

(Optional) Specifies when the job is scheduled to run in the home Application. The Application containing the external job does not have to be active when the job in the home Application completes. If the job in the home Application is (was) submitted at the scheduled time, the server submits the external job.

Examples: 8AM TODAY, TODAY

Note: This field resolves to a specific time. For example, TODAY resolves to midnight today, not anytime today.

From

(Optional) Specifies the start of an explicit range of time when the job is scheduled to run in the home Application. Use this field with the to field.

Example: NOW LESS 2 HOURS

to

(Optional) Specifies the end of an explicit range of time when the job is scheduled to run in the home Application. Use this field with the From field.

Example: NOW PLUS 2 HOURS

If the job in the home Application is submitted within the scheduled range of time, the server submits the external job in the distant Application. For example, using the preceding criteria, the server submits the external job in the distant Application if the job in the home Application is (was) submitted within 2 hours of the current time. When the job completes successfully in the home Application, the server posts the external job complete in the distant Application and releases its successors.

5. (Optional) Complete the following fields only if the external job runs on a different scheduler than the home Application:

   Agent name

   Specifies the name of the agent that you are using to route information between the schedulers.

   External scheduler name

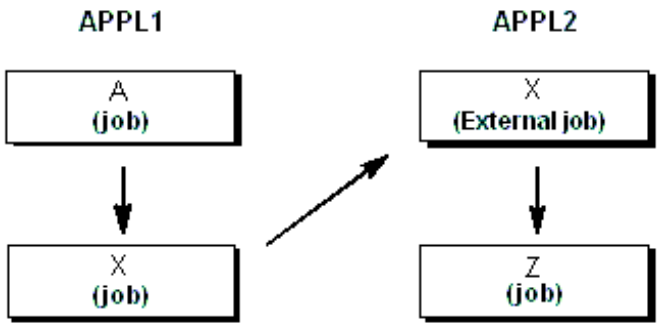
   Specifies the name of the other scheduler. This scheduler runs the home Application that submits the job that the external job depends on.

6. Click OK.

   The external job is defined.

Example: Synchronize Applications

Suppose that job X in APPL1 runs with an unpredictable frequency, and you want the server to complete external job X in APPL2, if job X in APPL1 ran in the past 24 hours or is scheduled to run in the next 24 hours. The following illustration displays the dependency between the two Applications:

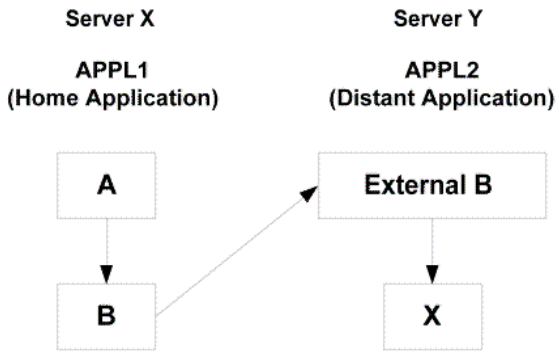


You would create two Applications as follows:

- APPL1
  - Set the run frequency of job A to run daily.
  - Make job X a successor of job A.
- APPL2
  - Define job X as an External-Same Scheduler job.
  - In job X, set the Home Application name field to APPL1, the From field to NOW LESS 24 HOURS, and the to field to NOW PLUS 24 HOURS.
  - Set the run frequency of job Z to run daily.
  - Make job Z a successor of external job X.

Example: Set Up a Dependency Between Two CA Workload Automation DE Systems

Suppose that job X defined in Server Y depends on job B defined in Server X. Server X runs the home Application to submit job B. The following illustration displays the dependency between the two CA Workload Automation DE systems.



You would create two Applications as follows:

- APPL1 on Server X
  - Make job B a successor of job A.
- APPL2 on Server Y
  - Define job B as an External-Other Scheduler job.
  - Set the Agent name field of job B to the name of the agent used to route information between the schedulers.
  - Set the External scheduler name field of job B to Server X.
  - Set the Home Application name field of job B to APPL1.
  - Make job X a successor of external job B.

Note: For more information on integrating two CA Workload Automation DE systems, see the *Implementation Guide*.

More information:

Cross-Application Dependencies (see page 552)

File Transfer Jobs

File Transfer jobs let you transfer ASCII, binary, and EBCDIC files between an agent computer and a remote computer. You can securely transfer binary files. The binary 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 File Transfer jobs:

FTP

Lets you transfer files using FTP.

SCP

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.

SFTP

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.

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.

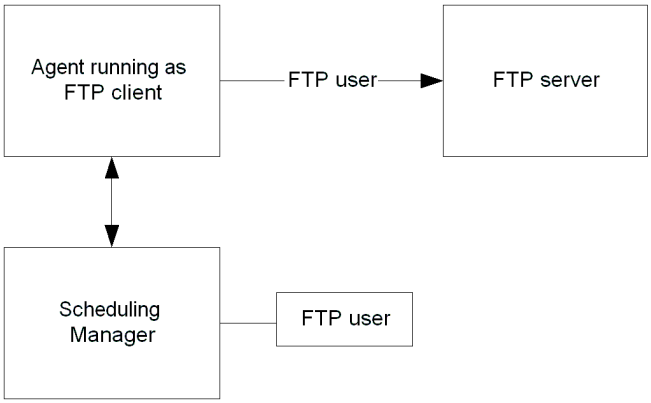
For the QSYS file system on i5/OS systems, you can only transfer members of FILE objects.

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

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.

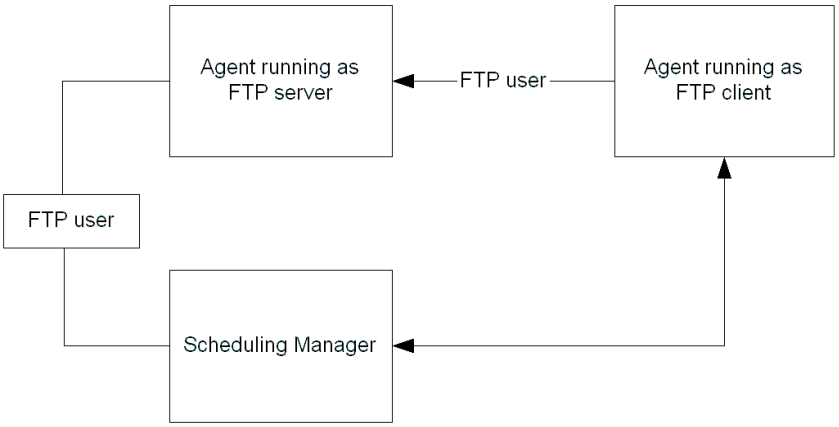
When the agent runs as an FTP client only, other FTP clients (such as other agents) cannot log in to the agent to transfer files. To let other FTP clients log in and transfer files, the agent administrator needs to set up the agent to run as an FTP server.

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.

Define an FTP Job

You can define an FTP 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the FTP job from the File Transfer group in the Palette view, and drag the job to the workspace.

   The FTP icon appears on the Application workspace view.

3. Right-click the FTP icon, and select Edit from the pop-up menu.

   The Basic page of the FTP dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent where the FTP transfer takes place.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   User

   Specifies the user ID of the user with the authority to download the file from the remote FTP server or upload the file to the remote FTP server. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Server address

Specifies the DNS name or IP address of a remote server.

Example: 172.24.36.107 (IPv4) or 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

Remote file name

Specifies the file's source location (if downloading) or the file's destination (if uploading).

Note: You can specify multiple files. Separate each file name with a semi-colon. The number of files specified in the Local file name and Remote file name fields must match.

UNIX/Windows:

- If you are uploading a file, you must 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.
- 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 cannot use wildcards in the path.
- On UNIX, if you want to use a Windows file as a remote file, you must use a forward slash at the beginning of the path statement and between the directories and file name, for example, /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`

Local file name

Specifies the file's destination (if downloading) or the file's source location (if uploading).

Note: You can specify multiple files. Separate each file name with a semi-colon. The number of files specified in the Local file name and Remote file name fields must match.

UNIX/Windows:

- If you are downloading a file, you must 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.
- 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 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

Transfer direction

Indicates the direction of transfer (Download or Upload).

Default: Download

Transfer code type

Specifies the type of data you are transferring. Options are as follows:

Binary

Indicates a binary transfer.

ASCII

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.

EBCDIC

Indicates an EBCDIC transfer.

Note: EBCDIC applies to jobs running on System i5 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.

Default: Binary

5. (Optional) Specify the following additional information:

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Server port

Specifies the port number of the remote server.

Default: 22

Local user

Specifies a user ID on the UNIX or Linux computer where the agent is installed. This user ID determines the access permissions of a downloaded file on the agent computer and does not apply to uploads. When the file is downloaded, the file is created with this user as the file owner. To set the owner of a downloaded file, the agent must run as root.

Notes:

- The local user does not need to be defined in the Topology.
- 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 value in this field overrides the default setting specified in the ftp.download.owner parameter in the agent's agentparm.txt.

Compression level

Defines the compression level (0 is no data compression, 9 is the best data compression).

Default: Default compression level set on the agent FTP client

SSL connection

Indicates whether to transfer the data using Secure Sockets Layer (SSL) communication (True or False).

Default: Default FTP setting (regular FTP or SSL FTP) on the agent

FTP site commands

Defines the commands that are to be executed prior to file transfer. You can use this section to send site-specific FTP commands to FTP servers.

6. Click OK.

   The FTP job is defined.

Example: Upload Multiple Files to a Directory Using a Wildcard

Suppose that you want to upload all the files in the C:\ca directory to the E:\ftp directory on a remote Windows server.

To upload multiple files to a directory using a wildcard

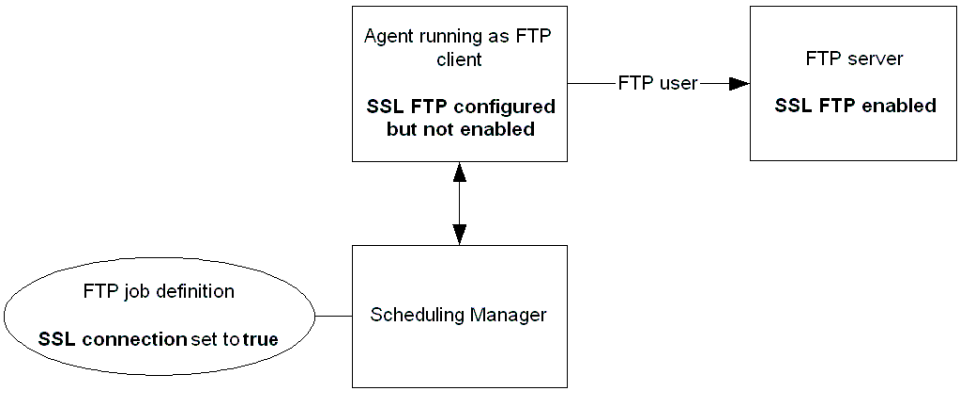
1. Enter the following information in the Basic page:
   - Name—FTP_UPLOAD
   - Agent name—SYSAGENT
   - User—test
   - Server address—winserver
   - Remote file name—E:\ftp
   - Local file name—C:\ca*
2. Select the Upload and ASCII option buttons.
3. Click OK.

Example: Upload a File From a Local Computer to a Remote Windows Server Using SSL FTP

Suppose that 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.

To securely upload a file from the agent FTP client to the remote Windows FTP server, set SSL connection to True in the FTP job definition. Although the agent FTP client does not have SSL FTP enabled, the file is uploaded using SSL FTP because the agent FTP client has SSL FTP *configured* and the FTP server has SSL FTP enabled.

The following diagram shows the scenario:



Suppose the FTP job FTP_UPLOAD uploads the file d:\files_to_upload\filename.txt from the agent FTP client to the c:\uploaded_files directory on a remote Windows server named winserver. The agent SYSAGENT logs into the remote Windows server using user user1. Because the FTP client has SSL configured but not enabled, SSL connection is set to True to transfer the file securely.

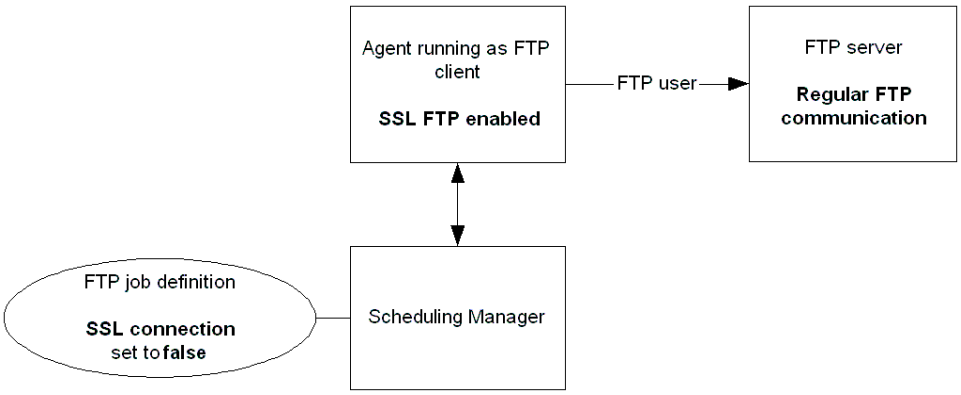
To upload a file from a local computer to a remote Windows server using SSL FTP

1. Enter the following information in the Basic page:
   - Name—FTP_UPLOAD
   - Agent name—SYSAGENT
   - User—user1
   - Server address—winserver
   - Remote file name—c:\uploaded_files\filename.txt
   - Local file name—d:\files_to_upload\filename.txt
2. Select the Upload and ASCII option buttons.
3. Select True from the SSL connection drop-down list.
4. Click OK.

Example: Download a File from a Remote UNIX Server that Does Not Support SSL FTP to a Local Computer that Supports SSL FTP

Suppose that 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). The remote UNIX server does not support SSL FTP.

The following diagram shows the scenario:



Suppose that the FTP job FTP_DOWNLOAD downloads the file /files_to_download/filename.txt from the remote UNIX server hpunix to the c:\downloaded_files directory on the local server. The agent SYSAGENT logs into the remote UNIX server using user user1. Because the FTP server does not support SSL FTP, SSL connection is set to False.

To download a file from a remote UNIX server that does not support SSL FTP to a local server that supports SSL FTP

1. Enter the following information in the Basic page:
   - Name—FTP_DOWNLOAD
   - Agent name—SYSAGENT
   - User—user1
   - Server address—hpunix
   - Remote file name—/files_to_download/filename.txt
   - Local file name—c:\downloaded_files\filename.txt
2. Select the Download and ASCII option buttons.
3. Select False from the SSL connection drop-down list.

   Note: To transfer FTP data, you must set SSL connection to False. Otherwise, the job will fail.
4. Click OK.

Example: Upload an ASCII-encoded File in the Root File System from an i5/OS System to a UNIX System

Suppose that the FTP job FTP_UPLOAD uploads a file named textfile in the root file system is uploaded from an i5/OS system to a UNIX system. Note that the two locations include a complete path statement.

To upload an ASCII-encoded file in the root file system from an i5/OS system to a UNIX system

1. Enter the following information in the Basic page:
   - Name—FTP_UPLOAD
   - Agent name—I5AGENT
   - User—test
   - Server address—hpunix
   - Server port—5222
   - Remote file name—/u1/qatest/ftpdata/textfile
   - Local file name—/home/cybesp/testfile
2. Select the Upload and ASCII option buttons.
3. Select False from the SSL connection drop-down list.

   Note: To transfer FTP data, you must set SSL connection to False. Otherwise, the job will fail.
4. Click OK.

Example: Download a QSYS.LIB EBCDIC-encoded File

Suppose that the FTP_DOWNLOAD job downloads an EBCDIC-encoded file named datafile in the QSYS.LIB file system from an i5/OS system to another i5/OS system. Note that the file names are specified in the path format.

To download a QSYS.LIB EBCDID-encoded file

1. Enter the following information in the Basic page:
   - Name—FTP_DOWNLOAD
   - Agent name—I5AGENT
   - User—test
   - Server address—i5os
   - Server port—5222
   - Remote file name—/QSYS.LIB/DATALIB.LIBDATAFILE.FILE/DATA.MBR
   - Local file name—/QSYS.LIB/ESPLIB.LIB/DOWNLOAD.FILE/DATA.MBR
2. Select the Download and EBCDIC option buttons.
3. Select False from the SSL connection drop-down list.

   Note: To transfer FTP data, you must set SSL connection to False. Otherwise, the job will fail.
4. Click OK.

Example: Upload and Convert an EBCDIC-encoded File to ASCII

Suppose that the FTP_CONVERT job uploads a file member named RESULT from an i5/OS system to a UNIX system. The job automatically detects that the RESULT file member is EBCDIC-encoded and that the target UNIX system accepts only ASCII-encoded files. The I5Agent agent is configured to automatically convert EBCDIC-encoded files to ASCII during an upload to an ASCII system, so the RESULT file member uploads successfully.

To upload and convert an EBCDIC-encoded file to ASCII

1. Enter the following information in the Basic page:
   - Name—FTP_CONVERT
   - Agent name—I5AGENT
   - User—test
   - Server address—hpunix
   - Server port—5222
   - Remote file name—/u1/qatest/ftpdata/resultup
   - Local file name—/QSYS.LIB/MYLIB.LIB/DATA.FILE/RESULT.MBR
2. Select the Upload and ASCII option buttons.
3. Select False from the SSL connection drop-down list.

   Note: To transfer FTP data, you must set SSL connection to False. Otherwise, the job will fail.
4. Click OK.

Example: Compress a File and Download it Using SSL

Suppose that the local server has an agent running as an FTP client. The remote server has the agent running as an FTP server. SSL FTP is enabled on both FTP client and FTP server. Both servers operate on a low bandwidth network.

Suppose that the FTP job FTPJOB downloads a large file named /files_to_download/largefile.txt from the remote server aixunix to the c:\downloaded_files directory on the FTP client. The agent SYSAGENT logs into the remote UNIX server using user user1. The servers are on a low bandwidth network, so the data is compressed at compression level 3. By default, the job runs using SSL FTP because SSL FTP is enabled on both FTP client and FTP server.

To compress a file and download it using SSL

1. Enter the following information in the Basic page:
   - Name—FTPJob
   - Agent name—SYSAGENT
   - User—user1
   - Server address—aixunix
   - Remote file name—/files_to_download/largefile.txt
   - Local file name—c:\downloaded_files\largefile.txt
   - Compression level (0-9)—3
2. Select the Download and ASCII option buttons.
3. Click OK.

Example: Send Two FTP Commands to an FTP Server

Suppose that you want to send two FTP commands to the FTP server prior to transferring a file.

To send two FTP commands to an FTP server

1. Enter the following information in the Basic page:
   - Name—FTP_SITECOMMANDS
   - Agent name—FTPAGENT
   - User—user1
   - Server address—ftp.example.com
   - Server port—21
   - Remote file name—/pub/cazip.exe
   - Local file name—/tmp/bla
   - Compression level (0-9)—8
2. Select the Download and EBCDIC option buttons.
3. Click Add in the FTP site commands section.

   A new row is added to the Command table.
4. Enter **site date** in the new row.
5. Click Add in the FTP site commands section.

   A new row is added to the Command table.
6. Enter **site recfm=FB** in the new row.
7. Click OK.

More information:

FTP Jobs (see page 223)
Using a Namespace for a User that has Different Passwords (see page 63)

Secure Sockets Layer (SSL) FTP Usage

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 connection value for an individual job overrides this default FTP setting on the agent FTP client as follows:

- If the SSL connection value is set to True, the data is transferred using SSL FTP.
- If the SSL connection value is set to False, the data is transferred using regular FTP.
- If the SSL connection value is not specified, the data is transferred using the default FTP setting (regular FTP or SSL FTP).

Note: You must set the SSL connection to False if the FTP client has SSL FTP enabled, but the FTP server does not. Otherwise, the job will fail.

Define an SCP Job

You can define an SCP 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SCP job from the File Transfer group in the Palette view, and drag the job to the workspace.

   The SCP icon appears on the Application workspace view.

3. Right-click the SCP icon, and select Edit from the pop-up menu.

   The Basic page of the SCP dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent where the secure transfer takes place.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Transfer direction

   Indicates the direction of transfer (Download or Upload).

   Default: Download

   Server address

   Specifies the DNS name or IP address of a remote server.

   Example: 172.24.36.107 (IPv4) or 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

   Remote directory

   Specifies the file's remote source directory (if downloading) or the file's remote destination directory (if uploading).

Remote file name

Specifies the file's source location (if downloading) or the file's destination (if uploading).

Note: Wildcards are not permitted.

Local file name

Specifies the file's destination (if downloading) or the file's source location (if uploading).

Note: Wildcards are not permitted.

User

Specifies the user ID of the user with the authority to download the file from the remote FTP server or upload the file to the remote FTP server. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

5. (Optional) Specify the following additional information:

Server port

Specifies the port number of the remote server.

Default: 22

Local user

Specifies a user ID on the UNIX or Linux computer where the agent is installed. This user ID determines the access permissions of a downloaded file on the agent computer and does not apply to uploads. When the file is downloaded, the file is created with this user as the file owner. To set the owner of a downloaded file, the agent must run as root.

Notes:

- The local user does not need to be defined in the Topology.
- 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 value in this field overrides the default setting specified in the ftp.download.owner parameter in the agent's agentparm.txt.

The remote os type

Specifies the remote operating system type in a secure file transfer (UNIX or Windows). The remote operating system type is used to determine the path separator on the remote system.

6. Click OK.

   The Secure Copy job is defined.

Example: Download a File Using the Secure Copy Protocol

Suppose that you want to download the install.log1 file from the root directory on the chi-linux server using the Secure Copy Protocol (SCP).

To download a file using the Secure Copy Protocol

1. Enter the following information in the Basic page:
   - Name—SCP_DOWNLOAD
   - Agent name—WINAGENT
   - Server address—chi-linux
   - Server port—22
   - Remote directory—/root
   - Remote file name—install.log
   - Local file name—C:\temp\install.log1
   - User—causer
2. Select the Download option button in the Transfer direction section.
3. Click OK.

Example: Change the Owner of a File Downloaded from a Remote Server

Suppose that you want to download 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.

To change the owner of a file downloaded from a remote server

1. Enter the following information in the Basic page:
   - Name—SCP1_D
   - Agent name—UNIXAGENT
   - Server address—simon
   - Server port—22
   - Remote directory—/u1/test/ftpdata
   - Remote file name—file_size8
   - Local file name—/ul/causer/data/scp_file_size8_local_user
   - User—causer
   - Local user—test
2. Select the Download option button in the Transfer direction section.
3. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

Define an SFTP Job

You can define an SFTP 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SFTP job from the File Transfer group in the Palette view, and drag the job to the workspace.

   The SFTP icon appears on the Application workspace view.

3. Right-click the SFTP icon, and select Edit from the pop-up menu.

   The Basic page of the SFTP dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent where the secure transfer takes place.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Transfer direction

   Indicates the direction of transfer (Download or Upload).

   Default: Download

   Server address

   Specifies the DNS name or IP address of a remote server.

   Example: 172.24.36.107 (IPv4) or 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

   Remote directory

   Specifies the file's remote source directory (if downloading) or the file's remote destination directory (if uploading).

Remote file name

Specifies the file's source location (if downloading) or the file's destination (if uploading). This field is not required if you are uploading multiple files.

Notes:

- For uploads, you must specify the file name without wildcards.
- For downloads, you can use wildcards for the file name. The asterisk (*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- If a wildcard is used in a 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 cannot rename files if wildcards are used.

Local file name

Specifies the file's destination (if downloading) or the file's source location (if uploading).

Notes:

- For downloads, you must specify the full path and file name without wildcards.
- For uploads, you can use wildcards for the file name. The asterisk (*) is a wildcard for zero or more characters and the question mark (?) is a wildcard for a single character.
- If a wildcard is used in a local file name for upload, the Remote file name field is not required. A wildcard transfer is equivalent to an mget transfer using an FTP client.
- You cannot rename files if wildcards are used.
- You cannot use wildcards in the path.

User

Specifies the user ID of the user with the authority to download the file from the remote FTP server or upload the file to the remote FTP server. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

5. (Optional) Specify the following additional information:

   Server port

   Specifies the port number of the remote server.

   Default: 22

   Local user

   Specifies a user ID on the UNIX or Linux computer where the agent is installed. This user ID determines the access permissions of a downloaded file on the agent computer and does not apply to uploads. When the file is downloaded, the file is created with this user as the file owner. To set the owner of a downloaded file, the agent must run as root.

   Notes:

   - The local user does not need to be defined in the Topology.
   - 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 value in this field overrides the default setting specified in the ftp.download.owner parameter in the agent's agentparm.txt.

   The remote os type

   Specifies the remote operating system type in a secure file transfer (UNIX or Windows). The remote operating system type is used to determine the path separator on the remote system.

6. Click OK.

   The Secure FTP job is defined.

Example: Upload a File Using the Secure File Transfer Protocol

Suppose that you want to upload the logs.tar file to the /u/tmp directory on the hpsupport server. The job uses the Secure File Transfer Protocol (SFTP).

Follow these steps:

1. Enter the following information in the Basic page:
   - Name—SFTP_UPLOAD
   - Agent name—WINAGENT
   - Server address—hpsupport
   - Remote directory—/u/tmp
   - Remote file name—logs.tar
   - Local file name—D:\temp\logs.tar
   - User—causer
2. Select the Upload option button in the Transfer direction section.
3. Click OK.

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 value in the Local file name field contains a wildcard, no value is specified in the Remote file name field.

Follow these steps:

1. Enter the following information in the Basic page:
   - Name—SFTP_UPLOAD_MULTIPLE
   - Agent name—WINAGENT
   - Server address—aixunix
   - Remote directory—/u1/build/uploaded
   - Local file name—c:\temp\upload*
   - User—causer
2. Select the Upload option button in the Transfer direction section.
3. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

Informatica Jobs

Informatica jobs let you start predefined workflows on Informatica PowerCenter. In Informatica, jobs are known as workflows. A workflow can contain multiple tasks. You can start the workflow from the beginning of the workflow or from a specified task. You can also start a workflow instance.

Informatica PowerCenter provides an environment that lets you load data into a centralized location, such as a data warehouse or operational data store (ODS). You can extract data from multiple sources, transform the data according to business logic you build in the client application, and load the transformed data into file and relational targets.

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

Define an Informatica Job

You can define an Informatica job to start a predefined workflow on Informatica PowerCenter. You can start the workflow from the beginning of the workflow or from a specified task. You can also start a workflow instance.

Notes:

- Due to a known issue in Informatica, you cannot run workflow instances in parallel.
- To run these jobs, your system requires CA WA Agent for Linux or Windows and CA WA Agent for Informatica.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Informatica job in the Palette view, and drag the job to the workspace.

   The Informatica icon appears on the Application workspace view.

3. Right-click the Informatica icon, and select Edit from the pop-up menu.

   The Basic page of the Informatica dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that runs the Informatica job.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Folder name

Specifies the Informatica PowerCenter repository folder name where the specified workflow resides. You can obtain this value from the Informatica server.

Limits: Up to 79 characters

Workflow name

Specifies the name of the Informatica PowerCenter workflow to be started.

Limits: Up to 79 characters

Note: You can type the workflow name or, if you are connected to the server, click the arrow search button next to the field to search for a workflow that is defined in Informatica PowerCenter. You can also view the sequence of tasks and the hierarchy of objects within a selected workflow.

5. (Optional) Specify the following additional information:

Informatica target

Specifies the name of the Informatica target (*infaTarget*). The corresponding directory (*install_dir*/config/informatica/*infaTarget*) stored on the agent contains the properties for connecting to the Informatica server and one or more database repositories.

Limits: Up to 128 characters

Note: If this field is not specified, the agent uses the default value specified in the informatica.server.target.default parameter in the agentparm.txt file.

Repository name

Specifies the name of the repository created on the Informatica server. The repository can be obtained from the Informatica Administrative console. The corresponding property file (*repositoryName*.properties) stored on the agent contains the database connection properties for a specified Informatica target (*infaTarget*).

Limits: Up to 128 characters

Note: If this field is not specified, the agent uses the default value specified in the informatica.repository.name.default parameter in the informatica.properties file stored on the agent.

Informatica user

Specifies the Informatica server user name. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: Up to 128 characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all of the user names that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- If this field is not specified, the agent uses the default value specified in the informatica.user.id.default parameter in the informatica.properties file stored on the agent.

Workflow instance

Specifies the name of the workflow instance to be started. When the task name and the workflow instance name are specified, the workflow instance is started from the specified task. When the task name is not specified, the workflow instance starts from the beginning. If the instance name is not specified, the workflow is started.

Limits: Up to 79 characters

Note: Before you select a workflow instance from the drop-down, you must specify the workflow in the Workflow name field. The workflow instances in the drop-down list are populated based on the specified workflow.

Task name

Specifies the name of the task to start the workflow from. If the task name is not specified, the workflow starts from the beginning.

Limits: Up to 79 characters

Note: You can type the task name or, if you are connected to the server, click the arrow search button next to the field to search for a task that is defined within the specified workflow. You can also view the details and the hierarchy of a selected task within the workflow.

Param file

Specifies the path and name of a parameter file on the Informatica server. Informatica PowerCenter processes the parameter file when running a workflow. The parameter file contains values for symbolic variables and must adhere to the format that Informatica PowerCenter expects. A property value that is set in a parameter file overrides a value for this property that is set at the session level or elsewhere.

Limits: Up to 256 characters

Pass on success only

Indicates whether the status of the workflow is retrieved directly from Informatica or inferred depending on the successful execution of all the tasks. Select one of the following values from the drop-down list:

- True—Returns the workflow status based on the status of tasks in the Informatica workflow. If all tasks in the workflow complete successfully, the agent reports the status of the workflow as successful. If any task in the workflow fails, the agent reports the status of the workflow as failed.
- False—Returns the workflow status as reported by Informatica. In Informatica, a workflow can succeed even if one or more tasks in the workflow fail.

Notes:

- If you do not select any value from the drop-down list, the agent uses the default value specified in the informatica.passonsuccessonly.default parameter in the agentparm.txt file.
- If no value is specified in the drop-down list or in the agentparm.txt file, the default value is False.

6. Click OK.

   The Informatica job is defined.

Example: Start a Workflow on Informatica PowerCenter from a Task

This example starts a workflow named WF_1 on Informatica PowerCenter from the copyfile task. The workflow is started from the copyfile task.

To start a workflow on Informatica PowerCenter from a task

1. Enter the following information on the Basic page of the job definition:
   - Name—INF_1
   - Agent name—AGENT_INF
   - Folder name—sandbox
   - Workflow name—WF_1
   - Task name—copyfile
2. Click OK.

Example: Start a Workflow with a Parameter File

Suppose that you want to start a workflow named WF_2 with the wf2.prm parameter file on Informatica server. When the workflow runs, Informatica PowerCenter processes the specified parameter file.

The wf2.prm parameter file includes symbolic variables that override the default log directory and file name:

```
;Workflow Parameters
[INFAPLUGIN.WF:wf_3]
;Session parameters
$PMSessionLogFile=wf3.log
$PMSessionLogDir=C:\Infa\Log
;mapping parameters
```

To start a workflow with a parameter file

1. Enter the following information on the Basic page of the job definition:
   - Name—INF_1
   - Agent name—AGENT_INF
   - Folder name—sandbox
   - Workflow name—WF_2
   - Param file—C:/Infa/wf2.prm
2. Click OK.

   The log files will be stored in the C:\Infa\Log directory on the Informatica server and their name will start with wf2.log. The actual files that Informatica generates will be longer, for example, wf2.log.1225.20131021131730.bin.

More information:

Informatica Jobs (see page 247)

Search for an Informatica Workflow

You can use the Search Informatica Workflows dialog to search for and select a workflow that is defined on Informatica PowerCenter. You can also view the sequence of tasks and the hierarchy of objects within a workflow.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the Informatica job definition.
3. Complete the following required fields:

   Agent name

   Specifies the name of the Informatica agent where the workflow runs.

   Folder name

   Specifies the Informatica PowerCenter repository folder name where the specified workflow resides.

4. Click the arrow search button next to the Workflow name field.

   The Search Informatica Workflows dialog opens.

   Note: The required fields in this dialog are populated with the information that is specified on the Basic page of the job definition. You can also select another repository folder from the Folder name drop-down list.

5. (Optional) Specify the information in the Informatica Server Info section.

   Note: The fields in this section are populated with the information that is specified on the Basic page of the job definition, if any.

6. (Optional) Specify the filter criteria in the following field:

   Matching pattern

   Specifies the matching pattern string that is used to retrieve the names of workflows. Only workflows whose names contain the search pattern are displayed. % is the only wildcard character that is supported for the pattern matching.

   Example: If you specify "A%ab%" as the matching pattern, all the workflows whose names start with A and contain ab in the middle or at the end are displayed. If you specify "%file%" as the matching pattern, all the workflows whose names contain file are displayed, for example: file_task, X_file_Task and T_file.

   Note: If you do not specify a matching pattern, all workflows that belong to the specified repository folder are displayed.

7. Click the arrow search button next to the Informatica Server Info section.

   The Workflows table is populated with the list of workflows that meet the matching pattern string.

   Note: You can sort the data in the table by clicking on a table column heading.

8. (Optional) Select a workflow and click Workflow Details to display the sequence of tasks for the selected workflow in a tabular format.

9. (Optional) Select a workflow and click Workflow Object Hierarchy to display the hierarchy of all tasks within the selected workflow in a graphical format.

10. Select a workflow and click Select Workflow to specify the workflow name in the job definition.

    The selected workflow is set on the Basic page of the job definition.

Example: List Workflows that Match a Specified Pattern

Suppose that you want to retrieve a list of Informatica workflows whose names contains INF and underscore (_).

To list workflows based on matching pattern criteria

1. Open the Basic page of the Informatica job definition.
2. Enter the following information:
   - Agent name—Agent_INF
   - Folder name—sandbox
3. Click the arrow search button next to the Workflow name field.

   The Search Informatica Workflows dialog opens.

4. Enter **%inf%_%** in the Matching pattern field.
5. Click the arrow search button next to the Informatica Server Info section.

   The Workflows table is populated with the list of workflows that meet the search criteria.

Search for a Task in an Informatica Workflow

You can use the Search Informatica Tasks dialog to search for and select a task that is defined in an Informatica workflow. When the workflow runs, it starts from the task that you select. By default, the workflow starts from the first task. You can also view the details and the hierarchy of a task within the workflow.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the Informatica job definition.
3. Complete the following required fields:

   Agent name

   Specifies the name of the agent that runs the Informatica job.

   Folder name

   Specifies the Informatica PowerCenter repository folder name where the specified workflow resides.

   Workflow name

   Specifies the name of the Informatica PowerCenter workflow to be started.
4. Click the arrow search button next to the Task name field.

   The Search Informatica Tasks dialog opens.

   Note: The required fields in this dialog are populated with the information that is specified on the Basic page of the job definition. You can also select another repository folder from the Folder name drop-down list and another workflow from the Workflow name drop-down list.
5. (Optional) Specify the information in the Informatica Server Info section.

   Note: The fields in this section are populated with the information that is specified on the Basic page of the job definition, if any.
6. Click the arrow search button next to the Informatica Server Info section.

   The Tasks table is populated with the workflow tasks that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.
7. (Optional) Select a task and click Task Details to display the details of the selected task in a tabular format.

   Note: You can only retrieve the details of a task that ran as part of the workflow at least once. If the task has not previously run, an error message is displayed.

8. (Optional) Select a task and click Workflow Object Hierarchy to display the hierarchy of the selected task within the workflow in a graphical format.
9. Select a task and click Select Task to specify the task name in the job definition.

   The selected task name is set on the Basic page of the job definition.

Example: Display All Tasks for a Workflow

Suppose that you want to display all tasks for a workflow named INF_job in the sandbox folder.

To display all tasks for a workflow

1. Open the Basic page of the Informatica job definition.
2. Enter the following information:
   - Agent name—Agent_INF
   - Folder name—sandbox
   - Workflow name—INF_job
3. Click the arrow search button next to the Task name field.

   The Search Informatica Tasks dialog opens.
4. Click the arrow search button next to the Informatica Server Info section.

   The Tasks table is populated with the tasks that are defined in the workflow.

Micro Focus Jobs

The Micro Focus Net Express JCL engine lets you run your mainframe JCL and COBOL programs in a Windows or UNIX environment.

You can schedule and control your Micro Focus jobs using your scheduling manager and CA WA Agent for Micro Focus. The agent runs on the same system as the Micro Focus Enterprise Server and provides an interface to the Micro Focus Net Express JCL engine.

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

Define a Micro Focus Job

You can define a Micro Focus job to run your mainframe JCL and COBOL programs in a UNIX or Windows environment.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Micro Focus job from the Palette view, and drag the job to the workspace.

   The Micro Focus icon appears on the Application workspace view.

3. Right-click the Micro Focus icon, and select Edit from the pop-up menu.

   The Basic page of the Micro Focus dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Micro Focus Agent that runs the job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Server

   Indicates whether the value specified in the Server name/address field is a server name or the IP address and port of the Micro Focus server.

   Default: Server name

   Server name/address

   Specifies the Micro Focus server name (Server name) or the IP address and port of the Micro Focus server (Server address). If you specify the server address, you must use the format tcp:*IP_address*:*port*.

   Example: tcp:99.9.9.99:2222

   Note: If the Micro Focus server runs as a service under the system account and the CA WA Agent for UNIX, Linux, or Windows runs under a particular user ID (not the system account), then you must specify the server address.

JCL

Indicates whether to pass the JCL by content or by reference. If you select the JCL content option button, the submitted JCL is read from the file specified in the JCL content/reference field, and the contents of the JCL file are physically sent to the server. If you select the JCL reference option button, only the name of the file specified in the JCL content/reference field is sent to the server. The server then reads and submits the named file. This option is more efficient, since passing the name of the file is more efficient than passing the contents of the JCL file.

Default: JCL content

Note: If you need to edit the job's JCL before submitting the job, or if the operator needs to be able to restart the job from a particular step, you must choose the JCL content option.

JCL content/reference

Specifies the path to and name of the file that the job runs.

Note: If you are connected to the CA Workload Automation DE server and the agent is running, you can use the JCL Content Browser to browse for the file.

5. (Optional) Click Other MF Parameters in the left pane.

   The Other MF Parameters page opens in the right pane.

6. Specify the following additional information:

   Micro focus user

   Specifies a Micro Focus user ID under whose authority the job runs. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

O/S user

Specifies an operating system user ID under whose authority the Micro Focus commands run. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Default: Operating system user ID on the system where the Micro Focus server and the agent are installed

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- If the agent runs as a service under the system account, the Micro Focus job is executed under the system account. If the Micro Focus server runs under a particular user ID (not the system account) and you specified a Micro Focus server name for the job, then you must specify the same operating system user ID for the job to find the server.

Version

Indicates the applicable environment (VSE/Power, JES2, or JES3).

7. Click OK.

   The Micro Focus job is defined.

Example: Define a Micro Focus Job

Suppose that you want a job named MFJOB to run the JCL located at c:\Program Files\micro focus\es-jcldemo\infinite.jcl using a Micro Focus server with IP address 172.16.10.10 and port 2222. This job runs on the MFAGENT agent computer.

To define a Micro Focus job

1. Enter the following information in the Basic page:
   - Name—MFJOB
   - Agent name—MFAGENT
2. Do the following in the Server section:
   a. Select the Server address option button.
   b. Enter **tcp:172.16.10.10:2222** in the Server name/address field.
3. Do the following in the JCL section:
   a. Select the JCL content option button.
   b. Enter **c:\Program Files\micro focus\es-jcldemo\infinite.jcl** in the JCL content/reference field.
4. Click OK.

Example: Run a Micro Focus Job Under a Different Micro Focus User ID

Suppose that you want to run a Micro Focus job under the authority of the Micro Focus user ID MFUSER01.

To run a Micro Focus job under a different Micro Focus user ID

1. Open the Micro Focus job for which you want to set the Micro Focus user ID.
2. Click Other MF Parameters in the left pane.

   The Other MF Parameters page opens in the right pane.
3. Enter **MFUSER01** in the Micro Focus user field.

   Note: The user ID and password must be defined in the Topology under the agent.
4. Click OK.

Example: Run a Micro Focus Job Under a Different Operating System User ID

Suppose that you want to run a Micro Focus job under the authority of the operating system user ID OSUSER.

To run Micro Focus jobs under a different operating system user ID

1. Open the Micro Focus job for which you want to set the operating system user ID.
2. Click Other MF Parameters in the left pane.

   The Other MF Parameters page opens in the right pane.
3. Enter **OSUSER** in the O/S user field.

   Note: The user ID and password must be defined in the Topology under the agent.
4. Click OK.

Example: Edit the JCL Before Submitting the Job

Suppose that you want to edit the Micro Focus job's JCL before submitting the job.

Note: To edit JCL, the agent must be set up as an FTP server. This is usually done by the agent administrator of the server where Micro Focus is installed. You can contact the administrator for the following information:

- Server
- Port
- User ID
- Password

To edit the JCL before submitting the job

1. Open the Micro Focus job for which you want to edit the JCL.
2. Click Edit JCL from Agent.

   Note: If the Edit JCL from Agent option is disabled, verify that the JCL content option button is selected.

   The FTP Logon dialog opens.

   Note: You only need to specify the FTP connection details on this dialog the first time you edit Micro Focus JCL. The client saves the details on the current server and reuses them the next time you use the same agent to edit the JCL.

   You can also add, edit, or delete an FTP connection from the main menu. Select Window, Preferences, Desktop Client, Define Perspective, Micro Focus FTP Connections and select an active server connection from the Server connections drop-down list.

3. Complete the required fields as appropriate, and click OK, if applicable.

   The Edit JCL dialog opens.

4. Edit the JCL contents and save the JCL with the same job name or use Save as to save with a different job name.

More information:

Micro Focus Jobs (see page 255)
Using a Namespace for a User that has Different Passwords (see page 63)

Pass Variables as Parameters to the JCL File

You can pass variables as parameters to the JCL file.

Follow these steps:

1. Open the Micro Focus job for which you want to pass variables as parameters to the JCL file.
2. Click JCL Variables in the left pane.

   The JCL Variables page opens in the right pane.

3. Click Add to pass a variable as a parameter to the JCL, and complete the following fields:

   Name

   Defines the name of the variable.

   Value

   Specifies a value for the variable.

4. (Optional) Repeat the previous step to pass additional variables as parameters to the JCL file.
5. Click OK.

   The variables are passed to JCL as parameters.

Example: Pass SORT and MAX as Parameters to the JCL file

In the following JCL, suppose that you want to pass SORT for symbolic variable VAR1 and MAX for symbolic variable VAR2:

```
//MFIDSAL1 JOB 'DAVINDER',CLASS=A,MSGCLASS=A,NOTIFY=MFIDSA

//*--------------------------------------------------------------
//*
//*--------------------------------------------------------------
//STEP1    EXEC PGM=%VAR1,PARM='%VAR2'
//STEPLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//         DD  DSN=MFIDSA.DSA.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*                        00510000
//CEEDUMP  DD  SYSOUT=*                        00520000
//SYSUDUMP DD  SYSOUT=*                        00530000
//OLDLIB   DD  DISP=(OLD),DSN=MFE.PROCLIB.L01
```

To pass SORT and MAX as parameters to the JCL file

1. Open the Micro Focus job for which you want to pass variables as parameters to the JCL file.
2. Click JCL Variables in the left pane.

   The JCL Variables page opens in the right pane.
3. Click Add.

   A new row is added to the JCL variables table.
4. Enter the following information in the table:
   - Name—VAR1
   - Value—SORT
5. Click Add.

   A new row is added to the JCL variables table.
6. Enter the following information in the table:
   - Name—VAR2
   - Value—MAX
7. Click OK.

   The CA WA Agent for Micro Focus submits the following JCL to the Micro Focus server when the job runs:

   ```
   //MFIDSAL1 JOB 'DAVINDER',CLASS=A,MSGCLASS=A,NOTIFY=MFIDSA

   //*--------------------------------------------------------------
   //*
   //*--------------------------------------------------------------
   //STEP1    EXEC PGM=SORT,PARM='MAX'
   ```

```
//STEPLIB  DD  DSN=CEE.SCEERUN,DISP=SHR
//         DD  DSN=MFIDSA.DSA.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*                        00510000
//CEEDUMP  DD  SYSOUT=*                        00520000
//SYSUDUMP DD  SYSOUT=*                        00530000
//OLDLIB   DD  DISP=(OLD),DSN=MFE.PROCLIB.L01
```

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.

File Trigger

Lets you monitor file activity and perform an action based on that activity.

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 Services Monitoring

Lets you monitor the status of Windows services.

Define a CPU Monitoring Job

You can define a CPU Monitoring 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.

You can use the job to do the following:

- Record the CPU usage at the time the job runs. In this case, the job runs only once.
- Monitor CPU usage and trigger an alert if the CPU usage meets the defined criteria. For example, if the computer is using between 80 and 100 percent of CPU, the job continues to run until forced complete.

Note: The CPU usage reported as greater than 100 percent occurs when a partitioned or virtualized machine has overcommitted its CPU allocation and was able to use unused CPU from the rest of the machine.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.
2. Select the CPU Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The CPU Monitoring icon appears on the Application workspace view.
3. Right-click the CPU Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the CPU Monitoring dialog opens.
4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors CPU usage.

Wait mode

Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

Return immediately

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes.

Note: When using this option, you must also specify a value in the From field, the To field, or both.

Continuous monitoring, using alert

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.

Note: When using this option, you must specify an alert in the text field or the job defaults to the Return immediately option. You must also specify a value in the From field, the To field, or both.

Range

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 fields. 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 fields.

Note: This field does not apply when the Wait mode field is set to Return immediately.

Monitor when CPU percentage

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.

Notes:

- The percentage of available and used CPU appears in the Status field in the Details dialog (accessible from the Monitor perspective).
- This field does not apply when the Wait mode field is set to Return immediately.

5. (Optional) Specify the following additional information:

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

From

Defines the lower boundary of CPU usage in percent.

Note: If you specify a value in the From field without specifying a value in the To field, the range is between the lower boundary and 100 percent.

To

Defines the upper boundary of CPU usage in percent.

Note: If you specify a value in the To field without specifying a value in the From field, the range is between zero percent and the upper boundary.

No change

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.

Notes:

- To use this field, specify an alert in the Continuous monitoring, using alert field.
- The alert will trigger only if the change value is within the range specified by the From and To fields and 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.

6. Click OK.

   The CPU Monitoring job is defined.

Example: Trigger Alerts Based on Available CPU

Suppose that you want an alert to trigger whenever available CPU is less than 25 percent or greater than 75 percent. Subsequent alerts are triggered when the CPU usage changes by more than 10 percent. This job runs on the default agent computer.

To trigger Alerts based on available CPU

1. Enter the following information in the Basic page:
   - Name—CPUMON
   - Agent name—AGENT
2. Select the Continuous monitoring, using alert option button in the Wait mode section, and enter **CPUMONITOR** in the text field.
3. Enter the following information in the remaining fields:
   - From—25
   - To—75
   - No change—10
4. Select the Outside option button in the Range section.
5. Select the Available option button in the Monitor when CPU percentage section.
6. Click OK.

Using this example, the following table shows when alerts would be triggered with and without the No change value:

Time	CPU	Is the alert triggered when No change is not specified?	Is the alert triggered when No Change is specified at 10 percent?
14:00:01	25 percent	No. Available CPU must be below 25 percent or above 75 percent.	No. Available CPU must be below 25 percent or above 75 percent.
14:00:02	20 percent	Yes. Available CPU is below 25 percent.	Yes. Available CPU is below 25%.
14:00:03	19 percent	Yes. Available CPU is below 25 percent.	No. Available CPU remains below 25 percent, but the change from the last reading is only 1 percent.
14:00:04	8 percent	Yes. Available CPU is below 25 percent.	Yes. CPU usage has changed 12 percent from the last time the alert was triggered.
14:00:05	19 percent	Yes. Available CPU is below 25 percent.	Yes. CPU usage has changed 11 percent from the last time the alert was triggered.
14:00:06	32 percent	No. Available CPU must be below 25 percent or above 75 percent.	No. Although CPU usage changed by more than 10 percent, it no longer falls within the range defined by the From and To fields. It is not below 25 percent or above 75 percent.

More information:

Continuous Monitoring Usage (see page 64)

Define a Disk Monitoring Job

On Windows and UNIX systems, you can define a Disk Monitoring 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.

For a newly created file system, on any platform, the operating system usually reports the file system overhead as used bytes.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Disk Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The Disk Monitoring icon appears on the Application workspace view.

3. Right-click the Disk Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the Disk Monitoring dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors the disk space.

   Wait mode

   Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

   Return immediately

   Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes.

Note: When using this option, you must also specify a value in the From field, the To field, or both.

Continuous monitoring, using alert

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.

Note: When using this option, you must specify an alert in the text field or the job defaults to the Return immediately option. You must also specify a value in the From field, the To field, or both.

Drive to monitor

Specifies the path to the disk, logical partition, or auxiliary storage pool to be monitored.

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 format /dev/QASP*digit1digit2*, where *digit1* and *digit2* indicate the ASP number.

Range

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 fields. 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 fields.

Note: This field does not apply when the Wait mode field is set to Return immediately.

Monitor when disk space

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.

Note: This field does not apply when the Wait mode field is set to Return immediately.

5. (Optional) Specify the following additional information:

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

From

Defines the lower boundary of disk usage. The unit for this value is specified in the Format field.

Note: If you specify a value in the From field without specifying a value in the To field, the range is between the lower boundary and the maximum available disk space.

To

Defines the upper boundary of disk usage. The unit for this value is specified in the Format field.

Note: If you specify a value in the To field without specifiying a value in the From field, the range is between and the upper boundary.

No change

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 field.

Notes:

- To use this field, specify an alert in the Continuous monitoring, using alert field.
- The alert will trigger only if the change value is within the range specified by the From and To fields and 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.

Format

Specifies the unit of measurement used to monitor available or used disk space. Options are the following:

Percent

Monitors disk usage by percentage. This is the default.

Note: If you select Percent, the values specified by the From and To fields 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.

6. Click OK.

The Disk Monitoring job is defined.

Example: Triggering Alerts Based on Used Disk Space

Suppose that a Disk Monitoring job named DISKMON continuously monitors the local disk C. If the used space falls within 50 percent and 100 percent, the server triggers the alert named DISK. The server triggers subsequent alerts when the used space changes by 5 percent or more, but remains within the 50 percent and 100 percent boundaries. The job does not complete unless you force it complete using the Monitor perspective. This job runs on the default agent computer.

To trigger Alerts based on used disk space

1. Enter the following information in the Basic page:
   - Name—DISKMON
   - Agent name—AGENT
2. Select the Continuous monitoring, using alert option button in the Wait mode section, and enter **DISK** in the text field.
3. Enter the following information in the remaining fields:
   - Drive to monitor—C
   - From—50
   - To—100
   - No change—5
4. Select Percent from the Format drop-down list.
5. Select the Within option button in the Range section.
6. Select the Used option button in the Monitor when disk space section.
7. Click OK.

More information:

Continuous Monitoring Usage (see page 64)

File Trigger Jobs

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

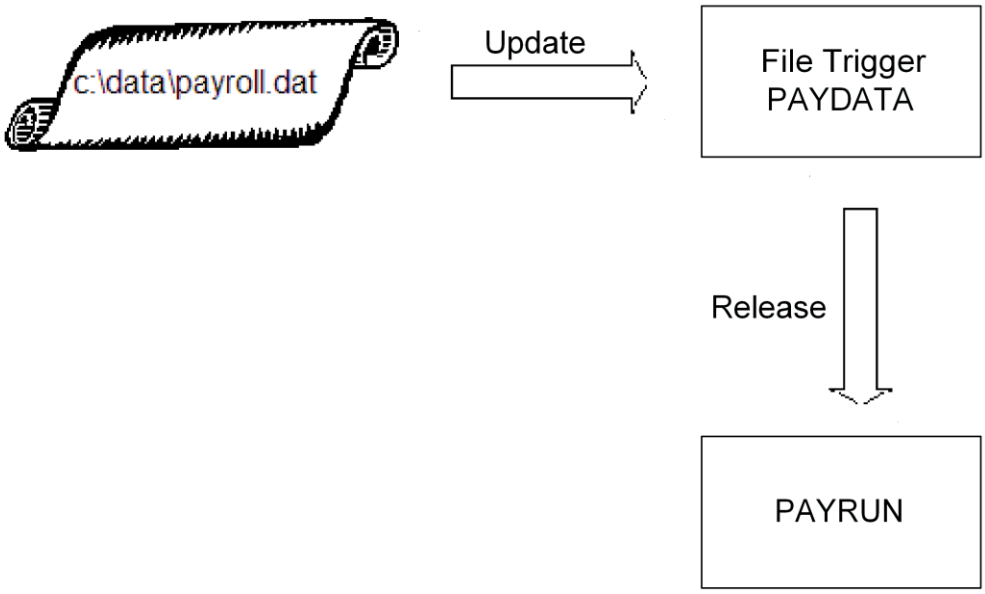
Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, Windows, or i5/OS.

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:



Polling Interval for File Trigger Jobs

File Trigger jobs monitor file activity using a polling interval, which is every 30 seconds by default. File Trigger jobs do not detect multiple updates during the polling interval. They also do not detect changes that cancel each other out.

For example, if the job monitors for updates to a file, and the file is updated twice during the polling interval, the trigger occurs only once for the two updates. If the job monitors for the creation of a file, and the file is created and deleted during the polling interval, the trigger does not occur. Because the file did not exist when the directory was polled, the job does not detect the file creation and deletion.

The agent administrator can change the number of seconds between polls by configuring the filemonplugin.sleepperiod parameter in the agentparm.txt file.

Note: For more information about the filemonplugin.sleepperiod parameter, see the agent *Implementation Guide* for your operating system.

Define a File Trigger Job

You can define a 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, or i5/OS.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the File Trigger job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The File Trigger icon appears on the Application workspace view.

3. Right-click the File Trigger icon, and select Edit from the pop-up menu.

   The Basic page of the File Trigger dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors the file activity.

File name

Specifies the path to and name of the file to monitor.

Note: If you are connected to the server and the agent is running, you can use the File Browser to browse for the file.

File trigger type

Indicates the file activity to monitor for (Created, Updated, Deleted, Expanded, Shrunk, Existing, and Non-Existing).

5. (Optional) Specify the following additional information:

Recursive

Monitors for file activity in the specified directory and all of its subdirectories.

When file reaches

Specifies the number of bytes the file size must reach to release the job's successors or trigger an Alert. This field is used with the Created, Expanded, or Shrunk file trigger type.

Default: 1 byte

Note: Some file systems will create files with extra space in anticipation of further file expansion requests, therefore, specifying exact byte counts will not always work.

Change size in bytes

Specifies the number of bytes the file size must change to release the job's successors or trigger an Alert. This field is used with the Expanded or Shrunk file trigger type.

Default: 1 byte

Percentage change

Specifies the percentage the file size must change to release the job's successors or trigger an Alert. This field is used with the Expanded or Shrunk file trigger type.

Default: 1 percent

Monitor continuously using

Specifies the name of the Alert that is triggered when the specified file activity occurs.

No changes for

Specifies the number of minutes the file must remain unchanged to satisfy the monitor condition.

Default: 0 minutes

6. (Optional) Click User/Group specifications in the left pane.

   The User/Group specifications page opens in the right pane.

7. Specify the following additional information:

   Owner user ID

   Specifies the UNIX user ID that owns the file to be monitored. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Group

   Specifies the UNIX group that owns the file to be monitored.

   Monitor as user

   Specifies the Windows user ID that monitors the file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

8. Click OK.

   The File Trigger job is defined.

Example: Release Jobs Based on File Creation

Suppose that you want the File Trigger job PAYDATA to release its successors when the c:\data\payroll.dat file is created on the SYSAGENT agent computer running Windows.

To release jobs based on file creation

1. Enter the following information in the Basic page:
   - Name-PAYDATA
   - Agent name-SYSAGENT
   - File name-c:\data\payroll.dat
2. Select the Created option button in the File trigger type section.
3. Click OK.

Example: Release Jobs Based on File Deletion

Suppose that you want the File Trigger job PAYDATA to release its successors when the c:\data\payroll.dat file is deleted on the SYSAGENT agent computer running Windows.

To release jobs based on file deletion

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—c:\data\payroll.dat
2. Select the Deleted option button in the File trigger type section.
3. Click OK.

Example: Release Jobs Based on File Size

Suppose that you want the File Trigger job PAYDATA to release its successors when the /data/test file has a file size of at least one byte on the SYSAGENT agent computer running UNIX.

To release jobs based on file size

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/data/test
2. Select the Expanded option button in the File trigger type section.
3. Select the When file reaches option button and enter **1** in this field.
4. Click OK.

Example: Release Jobs When a File Expands by a Certain Size

Suppose that you want the File Trigger job PAYDATA to release its successors when the /credit/record file expands by at least 2 megabytes on the SYSAGENT agent computer running UNIX.

To release jobs when a file expands by a certain size

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/credit/record
2. Select the Expanded option button in the File trigger type section.
3. Select the Change size in bytes option button and enter **2097152** in this field.

   Note: The size must be entered in bytes (2 x 1024 bytes x 1024k = 2097152).
4. Click OK.

Example: Release Jobs When a File Shrinks by a Certain Percentage

Suppose that you want the File Trigger job PAYDATA to release its successors when the /amount/test file shrinks in size by 35 percent or more on the SYSAGENT agent computer running UNIX.

To release jobs when a file shrinks by a certain percentage

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/amount/test
2. Select the Shrunk option button in the File trigger type section.
3. Select the Percentage change option button and enter **35** in this field.
4. Click OK.

Example: Release Jobs When a File Shrinks to a Certain Size

Suppose that you want the File Trigger job PAYDATA to release its successors when the /cash/items/distribute file shrinks to less than 1000 bytes on the SYSAGENT agent computer running UNIX.

To release jobs when a file shrinks to a certain size

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/cash/items/distribute
2. Select the Shrunk option button in the File trigger type section.
3. Select the When file reaches option button and enter **1000** in this field.
4. Click OK.

Example: Release Jobs When a File Shrinks by a Certain Amount

Suppose that you want the File Trigger job PAYDATA to release its successors when the /cost/cash file shrinks by 10 bytes on the SYSAGENT agent computer running UNIX.

To release jobs when a file shrinks by a certain amount

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/cost/cash
2. Select the Shrunk option button in the File trigger type section.
3. Select the Change size in bytes option button and enter **10** in this field.
4. Click OK.

Example: Release Jobs When a File Reaches a Certain Size and Remains Unchanged for a Certain Number of Minutes

Suppose that you want a File Trigger job to release its successors when the /research/analysis file reaches a size of 100 bytes or more on the SYSAGENT agent computer, provided that the file size remains unchanged for 5 minutes or more.

To release jobs when a file reaches a certain size and remains unchanged for a certain number of minutes

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/research/analysis
2. Select the Expanded option button in the File trigger type section, select the When file reaches option button, and enter the following information:
   - When file reaches—100
   - No changes for—5
3. Click OK.

Using this example, the following table shows when the job completes with and without the No Changes for field value:

Time	File Size	Does the Job Complete When No Changes for is Not Specified?	Does the Job Complete When No Changes for is Specified at 5 Minutes?
14:00:00	56	No. File size is less than 100 bytes	No. File size is less than 100 bytes
14:05:00	56	No. File size is less than 100 bytes	No. Although file size has not changed in five minutes, it is less than 100 bytes
14:09:59	101	Yes. File size is more than 100 bytes	No. Although file size is greater than 100 bytes, it has changed in the last five minutes
14:14:59	101		Yes. File size is greater than 100 bytes and has not changed for five minutes

Example: Monitor the Same File With Multiple File Triggers

Suppose that two File Trigger jobs continuously monitor the size of the same file (/data/totals) on the SYSAGENT agent computer. The first job (SURVEY.EXPAND) triggers an Alert named MGT every time the file expands by at least 10 KB from its initial size. The second job (SURVEY.SHRINK) triggers an Alert named MGT every time the file shrinks by at least 10 KB from its initial size.

To monitor the same file with multiple file triggers

1. Enter the following information in the Basic page of the job SURVEY.EXPAND:
   - Name—SURVEY
   - Qualifier—EXPAND
   - Agent name—SYSAGENT
   - File name—/data/totals
2. Select the Expanded option button in the File trigger type section, select the Change size in bytes option button, and enter the following information:
   - Change size in bytes—10240

     Note: The size must be entered in bytes (10 x 1024 bytes = 10240).
   - Monitor continuously using—MGT
3. Click OK.
4. Enter the following information in the Basic page of the job SURVEY.SHRINK:
   - Name—SURVEY
   - Qualifier—SHRINK
   - Agent name—SYSAGENT
   - File name—/data/totals
5. Select the Shrunk option button in the File trigger type section, select the Change size in bytes option button, and enter the following information:
   - Change size in bytes—10240

     Note: The size must be entered in bytes (10 x 1024 bytes = 10240).
   - Monitor continuously using—MGT
6. Click OK.

The following table displays the triggers that occur if the initial size of the /data/totals file is 100 KB and it changes as follows:

File Size	Is the Alert Triggered for SURVEY.EXPAND?	Is the Alert Triggered for SURVEY.SHRINK?
80 KB	No. File size is less than its initial size.	Yes. File size has decreased 20 KB from its initial size.
90 KB	No. File size remains less than its initial size.	No. File size has increased.
110 KB	Yes. File size has increased 10 KB from its initial size.	No. File size has increased.
50 KB	No. File size is less than its initial size.	Yes. File size has decreased 50 KB from its initial size.
60 KB	No. File size remains less than its initial size.	No. File size has increased.

Example: Release Jobs if a File Exists

Suppose that you want a File Trigger job to release its successors if the /bank/account/money file exists on the SYSAGENT agent computer running on UNIX.

To release jobs if a file exists

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/bank/account/money
2. Select the Existing option button in the File trigger type section.
3. Click OK.

Example: Release Jobs if a File Does Not Exist

Suppose that you want the File Trigger job PAYDATA to release its successors if the /start/term/vacation file does not exist on the SYSAGENT agent computer running on UNIX.

To release jobs if a file does not exist

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—/start/term/vacation
2. Select the Non-Existing option button in the File trigger type section.
3. Click OK.

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

Suppose that a File Trigger job PAYDATA monitors for an update to the payroll.dat file on a remote Windows computer named CYBNT. jsmith is a user ID on CYBNT that has access to the AccountingFiles directory and is defined in the Topology under the SYSAGENT agent computer.

To run a File Trigger job that monitors a file on a remote Windows computer, you must do the following:

- Ask your CA Workload Automation DE administrator to define a user ID in the Topology that has access to the file on the remote Windows computer.
- Specify the file name using UNC (Universal Naming Convention) in the job definition.
- Specify the user ID in the job definition.

To monitor for an update to a file on a remote Windows computer

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—SYSAGENT
   - File name—\\CYBNT\AccountingFiles\payroll.dat
2. Select the Updated option button in the File trigger type section.
3. Enter **jsmith** in the Monitor as user field in the User/Group specifications page.
4. Click OK.

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

Suppose that you want the File Trigger job PAYDATA to release its successors if the /data/payroll.dat file is created on the UNIXAGENT agent computer and the file is owned by jdoe. The following may take place:

- If the file does not exist when the job is readied, the trigger does not occur until the file is created.
- If the file exists and the owner of the file is jdoe, when the job is readied, the trigger occurs immediately.
- If the file exists or is created, but the owner of the file is not jdoe, the file trigger does not complete. It waits until all of the specified criteria are satisfied, including the Owner user ID criteria. If the owner of the file is changed to jdoe, the file trigger completes.

To monitor for the creation of a file that is owned by a specified UNIX user ID

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—UNIXAGENT
   - File name—/data/payroll.dat
2. Select the Created option button in the File trigger type section.
3. Enter **jdoe** in the Owner user ID field in the User/Group specifications page.
4. Click OK.

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

Suppose that you want the File Trigger job PAYDATA to release its successors if the /data/payroll.dat file exists on a UNIX computer and the file is owned by the group accts. The following may take place:

- If the file exists and is owned by the group accts, when the job is readied, the trigger completes successfully.
- If the file exists and is not owned by the group accts, when the job is readied, the file trigger fails.

To monitor for the existence of a file that is owned by a specified UNIX group

1. Enter the following information in the Basic page:
   - Name—PAYDATA
   - Agent name—UNIXAGENT
   - File name—/data/payroll.dat
2. Select the Existing option button in the File trigger type section.
3. Enter **accts** in the Group field in the User/Group specifications page.
4. Click OK.

Example: Use Wildcards to Specify a File Name on the Root File System

Suppose that you want to monitor continuously for files that are created in the /home/cybesp/ directory in an i5/OS root file system. A File Trigger job, JOBDATA, monitors the file system continuously and triggers an Alert named A123 every time a file is created with a file name that matches the following criteria:

- Starts with PID
- Ends with four characters
- Has any extension

To use wildcards to specify a file name on the root file system

1. Enter the following information in the Basic page:
   - Name—JOBDATA
   - Agent name—I5AGENT
   - File name—/home/cybesp/PID????.*
   - Monitor continuously using—A123
2. Select the Created option button in the File trigger type section.
3. Click OK.

Example: Use a Generic Name to Specify a QSYS.LIB File Object

Suppose that you want the File Trigger job JOBDATA to release its successors when any file object with a file name that starts with PAY and ends with any characters is created in the QSYS.LIB file system.

To use a generic name to specify a QSYS.LIB file object

1. Enter the following information in the Basic page:
   - Name—JOBDATA
   - Agent name—I5AGENT
   - File name—'LIB/PAY*/*FILE'

     Note: The file name is enclosed in single quotation marks so that the text following /* is not interpreted as a comment. To only monitor the total size of the file object's members, specify *ALL as the member name.
2. Select the Created option button in the File trigger type section.
3. Click OK.

Note: The file name is enclosed in single quotation marks so that the text following /* is not interpreted as a comment.

Example: Monitor the size of a QSYS.LIB file object

Suppose that a File Trigger job named EXPMON is triggered when the EXPOBJ file object in the QSYS.LIB 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.

To monitor the size of a QSYS.LIB file object

1. Enter the following information in the Basic page:
   - Name—EXPMON
   - Agent name—I5AGENT
   - File name—'LIB/EXPOBJ/*FILE'

     Note: The file name is enclosed in single quotation marks so that the text following /* is not interpreted as a comment. To only monitor the total size of the file object's members, specify *ALL as the member name.
2. Select the Expanded option button in the File trigger type section.
3. Select the Change size in bytes option button and enter **10** in the field.
4. Click OK.

More information:

Continuous Monitoring Usage (see page 64)
Using a Namespace for a User that has Different Passwords (see page 63)
File Trigger Jobs (see page 274)

Define an IP Monitoring Job

You can define an IP Monitoring 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the IP Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The IP Monitoring icon appears on the Application workspace view.

3. Right-click the IP Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the IP Monitoring dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors the IP address or port.

IP address

Specifies the DNS name or IP address.

Example: 172.24.36.107 (IPv4) or 0:0:0:0:0:FFFF:192.168.00.00 (IPv6)

Status type

Indicates the status of the device at the IP address to be monitored (Running or Stopped).

Default: Stopped

Wait mode

Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

Return immediately

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes.

5. (Optional) Specify the following additional information:

Job class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

IP port

Specifies the port number for the IP address being monitored. The agent attempts to connect to this port.

6. Click OK.

The IP Monitoring job is defined.

Example: Monitor a Device at a Specific IP Address and Port Number

Suppose that an IP Monitoring job named IPMON monitors a device, such as an agent, at IP address 10.1.1.20 and port number 7510. If the device is currently running, the job completes successfully; otherwise, it fails. This job runs on the default agent computer.

To monitor a device at a specific IP address and port number

1. Enter the following information in the Basic page:
   - Name—IPMON
   - Agent name—AGENT
   - IP address—10.1.1.20
   - IP port—7510
2. Select the Running option button in the Status type section.
3. Select the Return immediately option button in the Wait mode section.
4. Click OK.

Define a Process Monitoring Job

You can define a Process Monitoring 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.
2. Select the Process Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The Process Monitoring icon appears on the Application workspace view.
3. Right-click the Process Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the Process Monitoring dialog opens.
4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that monitors the process execution.

Process name

Specifies the name of the process to be monitored.

UNIX: Specify a PID 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.

jobname

Specifies the job name on the i5/OS system. The value can be a generic name or *ALL.

- 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.

Status type

Indicates the status of the process to be monitored (Running or Stopped).

Default: Stopped

Wait mode

Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

Return immediately

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes.

5. (Optional) Specify the following additional information:

   Job class

   Specifies the job class under which this job runs. 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.

   Example: foo

   Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click OK.

   The Process Monitoring job is defined.

Example: Monitor the Agent Process

Suppose that a Process Monitoring job named AGMON monitors the process cybAgent.exe. If the process is running, the job continues monitoring the process until it stops. When the process stops, the job completes successfully. This job runs on the default agent computer.

To monitor the agent process

1. Enter the following information in the Basic page:
   - Name—AGMON
   - Agent name—AGENT
   - Process name—cybAgent.exe
2. Select the Stopped option button in the Status type section.
3. Select the Return at first occurrence option button in the Wait mode section.
4. Click OK.

Define a Text File Reading and Monitoring Job

You can define a Text File Reading and Monitoring 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Text File Reading and Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The Text File Reading and Monitoring icon appears on the Application workspace view.

3. Right-click the Text File Reading and Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the Text File Reading and Monitoring dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors the text file for a text string.

Text file name

Specifies the path to and name of the text file to search.

Note: If you are connected to the server and the agent is running, you can use the File Browser to browse for the file.

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*)

Text string

Defines the text string to search for. You can specify the text string as a regular expression.

Notes:

- 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.
- Quotation marks in the text string are taken literally. Do not specify quotes in the text string unless you want to search for a string containing quotes. For example, "text" and text are two different search strings.

Exists

Specifies whether the job monitors if a text string exists or does not exist.

Default: The check box is selected. If the text string exists, the job completes successfully, or an alert is triggered (if monitoring continuously).

Note: You can clear the Exists check box to monitor 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. To use this setting, you must select the Return immediately option button in the Other Monitor Parameters page.

Search mode

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 selected, the search boundaries defined by the First line and Last line fields in the Other Monitor Parameters page are numeric. This is the default.

Regular expressions

Searches for a text string within boundaries defined by regular expressions. If Regular expressions is selected, the search boundaries defined by the First line and Last line fields in the Other Monitor Parameters page are regular expressions.

Date/Time

Searches for a text string within boundaries defined by a specified date and time. If Date/Time is selected, a value must also be specified in the Time format field in the Other Monitor Parameters page.

5. (Optional) Specify the following additional information:

File encoding

Specifies the name of the character set used to encode the data in the file.

Examples: ISO-8859-1, UTF-8, UTF-16, UTF-16BE, UTF-16LE, US-ASCII

Note: If the character set of the file does not appear in the drop-down list, you can type the character set in the field. The supported character sets depends on your operating system and Java Virtual Machine (JVM).

6. (Optional) Click Other Monitor Parameters in the left pane.

The Other Monitor Parameters page opens in the right pane.

7. (Optional) Specify the following additional fields:

Wait mode

Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

Return immediately

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes. To use this option, the Exist check box in the Basic page must be selected.

Continuous monitoring, using alert

Specifies the identifier of an Aert 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. To use this option, the Exist check box in the Basic page must be selected.

Note: When using this option, you must specify an Alert in the text field or the job defaults to the Return immediately option. 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.

Job class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

First line

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:

- If you selected the Line search mode in the Basic page, specify the starting line number.
- If you selected the Regular expressions search mode in the Basic page, specify a regular expression. The format is case-sensitive. 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.
- If you selected the Date/Time search mode in the Basic page, specify a time stamp using the format defined in the Time format field. The format is case-sensitive.

Default: First line of the text file

Notes:

- If you do not specify a value in this field, you must specify a value in the Last line field. The job starts searching from the first line of the text file to the upper boundary specified by the Last line field.
- 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.

Last line

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:

- If you selected the Line search mode in the Basic page, specify the last line number.
- If you selected the Regular expressions search mode in the Basic page, specify a regular expression. The format is case-sensitive. 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.
- If you selected the Date/Time search mode in the Basic page, specify a time stamp using the format defined in the Time format field. The format is case-sensitive.

Default: Last line of the text file.

Notes:

- If you do not specify a value in this field, you must specify a value in the First line field. The job searches from the lower boundary specified by the first line field to the last line of the text file.
- To use this field, you must select the Return immediately option button.
- 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.

Time format

Defines the date and time pattern to use when the upper and lower boundaries are specified as date and time. The upper and lower boundaries are used to search inside a log file.

Time position

Specifies the first column of the time stamp in the log file.

Limit: Five digits

8. Click OK.

The Windows Text File Reading and Monitoring job is defined.

Example: Monitor a File for a String in the First 200 Lines of the File

Suppose that an Application contains a Text File Reading and Monitoring job (MONITORERROR) and a UNIX job. If the Text File Reading and Monitoring job completes successfully, the UNIX job is released.

The Text File Reading and Monitoring job monitors a file named c:\log_files\log.txt on the VLAGENT agent computer and searches for the text string, ERROR, in the first 200 lines of the file. If the string is not found, the job completes successfully and the UNIX job is released. If the string is found, the job fails and the UNIX job is not released.

To monitor a file for a string in the first 200 lines of the file

1. Enter the following information in the Basic page:
   - Name—MONITORERROR
   - Agent name—VLAGENT
   - Text file name—c:\log_files\log.txt
   - Text string—ERROR
2. Clear the Exists check box.
3. Select the Line option button in the Search mode section.
4. Enter the following information in the Other Monitor Parameters page:
   - First line—1
   - Last line—200
5. Select the Return immediately option button in the Wait mode section.
6. Click OK.

Example: Monitor a text file encoded in ISO-8859-1

Suppose that you want to monitor a text file that contains data encoded in the ISO Latin Alphabet No. 1 (also named ISO-LATIN-1). The job checks the text file immediately and completes successfully if the specified string is found. If the string is not found, the job fails.

To monitor a text file encoded in ISO-8859-1

1. Enter the following information in the Basic page:
   - Name—TEXTFILE_JOB
   - Agent name—MONAGT
   - Text file name—/export/home/logs/transactions.log
   - Text string—ERROR MESSAGE
2. Select ISO-8859-1 from the File encoding drop-down list.
3. Select the Line option button in the Search mode section.
4. Enter the following information in the Other Monitor Parameters page:
   - First line—1
   - Last line—50
5. Select the Return immediately option button in the Wait mode section.
6. Click OK.

More information:

Continuous Monitoring Usage (see page 64)

Time Format and Position Specifications

To limit the search in the text file to a time range, you can specify the time stamp's format and position in that file. The scheduling manager uses the time pattern as a mask for searching for that particular time stamp. In the time pattern, all ASCII letters are pattern letters. The following table displays the symbols used to create a time pattern string:

Symbol	Definition	Type	Example
G	Era designator	Text	AD
y	Year	Number	2000
M	Month in year	Text or number	July or 07
d	Day in month	Number	10

Symbol	Definition	Type	Example
h	Hour in AM/PM (1 ~ 12)	Number	12
H	Hour in day (0 ~ 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
D	Day in year	Number	189
F	Day of week in month	Text or number	2 (2 Wednesday July)
w	Week in year	Number	27
W	Week in month	Number	2
a	AM/PM marker	Text	PM
k	Hour in day (1 ~ 24)	Number	0
K	Hour in AM/PM (0 ~ 11)	Number	0
z	Time zone	Text	EST (Eastern Standard Time)
'	A delimiter to identify text strings. Any text enclosed in single quotation marks is treated as text rather than as time-format definition variables.		
''	Used to indicate single quotation		

Example: Sample Time Format Patterns

The following table displays sample time format patterns for July 10, 2006 at 12:08 p.m. Eastern Standard Time:

Format Pattern	Result
"yyyy.MM.dd G 'at' hh:mm:ss z"	2006.07.10 AD at 12:08:56 EST
"EEE MMM dd HH:mm:ss.SSS zzz yyyy"	Mon Jul 10 23:59:59.999 EST 2006
"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"	2006.July.10 AD 12:08 PM

Define a Windows Event Log Monitoring Job

You can define a Windows Event Log Monitoring 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Windows Event Log Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The Windows Event Log Monitoring icon appears on the Application workspace view.

3. Right-click the Windows Event Log Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the Windows Event Log Monitoring dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that monitors the event log.

Event log name

Specifies the name of the event log. Windows operating systems record events in three kinds of logs:

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..

5. (Optional) Specify the following additional information:

Event type

Indicates the event type to monitor in the event log (Error, Warning, Info, Audit success, and Audit failure).

Event ID

Specifies the event IDs to monitor.

Note: You must select a comparison operator (Less than, Less equal, Equal, Greater than, or Greater equal) to monitor individual event IDs, such as IDs greater than or equal to 500.

Monitor from date/time

Specifies the date and time of the Windows event log. The job monitors for an event log that occurs on or after the specified date and time. Specify the following format:

yyyymmdd hh:mm:ss

yyyy

Specifies the four-digit year.

mm

Specifies the two-digit month.

dd

Specifies the two-digit day.

hh

Specifies the two-digit hour.

mm

Specifies the two-digit minute.

ss

Specifies the two-digit second.

Example: 20090501 06:30:00

6. Click Other Monitor Parameters in the left pane.

   The Other Monitor Parameters page opens in the right pane.

7. (Optional) Specify the following additional information:

   Wait mode

   Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

   Return immediately

   Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

   Return at first occurrence

   Waits for the conditions to occur. When the conditions are met, the job completes.

Continuous monitoring, using alert

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.

Note: When using this option, you must specify an Alert in the text field or the job defaults to the Return immediately option.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Event source

Specifies the event source as displayed in the Windows Event Viewer.

Computer name

Specifies the local computer name where the event log is monitored.

Event category

Specifies the event category as displayed in the Windows Event Viewer.

Event description

Specifies the event description as displayed in the Windows Event Viewer.

8. Click OK.

The Windows Event Log Monitoring job is defined.

Example: Monitor a Security Log for Audit Failure Events

Suppose that a Windows Event Log Monitoring job named MONITORJOB monitors a security log for audit failure events that have IDs greater than or equal to 500. When such an event occurs in the security log, the job completes successfully. This job runs on the default agent computer.

To monitor a security log for audit failure events

1. Enter the following information in the Basic page:
   - Name—MONITORJOB
   - Agent name—AGENT
   - Event log name—Security
2. Select Audit failure in the Event type drop-down list.
3. Select Greater equal from the Event ID drop-down list and enter **500** in the adjacent text box.
4. Open the Other Monitor Parameters page.
5. Select the Return at first occurrence option button in the Wait mode section.
6. Click OK.

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

Suppose that a Windows Event Log Monitoring job named MONITORLOGJOB monitors an application log that occurs any time on or after May 1, 2005. When such an event occurs in the application log, the job completes successfully. This job runs on the default agent computer.

To monitor an application log that occurs on or after a specified date

1. Enter the following information in the Basic page:
   - Name—MONITORLOGJOB
   - Agent name—AGENT
   - Event log name—Application
   - Monitor from date/time—20050501 00:00:00
2. Open the Other Monitor Parameters page.
3. Select the Return at first occurrence option button in the Wait mode section.
4. Click OK.

More information:

Continuous Monitoring Usage (see page 64)

Define a Windows Service Monitoring Job

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

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Windows Service Monitoring job from the Monitoring group in the Palette view, and drag the job to the workspace.

   The Windows Service Monitoring icon appears on the Application workspace view.

3. Right-click the Windows Service Monitoring icon, and select Edit from the pop-up menu.

   The Basic page of the Windows Service Monitoring dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors the Windows service.

   Service name

   Specifies the name of the local Windows service to be monitored.

   Status type

   Indicates the status of the process to be monitored (Running, Stopped, Continue Pending, Pause Pending, Paused, Start Pending, Stop Pending, Exists, or Does not exist).

   Default: Running

   Note: On UNIX and i5/OS, only the Running and Stopped options apply.

Wait mode

Specifies whether the job waits until the monitor conditions are met or tries to verify them immediately. Options are the following:

Return immediately

Checks for the conditions immediately. If the conditions are met, the job completes successfully. If the conditions are not met, the job fails. This is the default.

Return at first occurrence

Waits for the conditions to occur. When the conditions are met, the job completes.

5. (Optional) Specify the following additional information:

   Job Class

   Specifies the job class under which this job runs. 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.

   Example: foo

   Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

6. Click OK.

   The Windows Service Monitoring job is defined.

Example: Monitor a Service for a Particular Status

Suppose that a Windows Service Monitoring job named SERVICEMON monitors the service named CA_Workload_Automation_7599. The job completes if the service status becomes “Continue Pending”. This job runs on the default agent computer.

To monitor a service for a particular status

1. Enter the following information in the Basic page:
   - Name—SERVICEMON
   - Agent name—AGENT
   - Service name—CA_Workload_Automation_7599
2. Select Continue Pending from the Status type drop-down list.
3. Select the Return at first occurrence option button in the Wait mode section.
4. Click OK.

MSSQL Server Jobs

MSSQL Server jobs let you execute predefined jobs in the Microsoft SQL Server Agent to perform administrative tasks. The jobs can be Transact-SQL scripts, command prompt applications, Microsoft ActiveX scripts, Integration Services packages, Analysis Services commands and queries, or Replication tasks.

Notes:

- To run these jobs, your system requires CA WA Agent for Windows and CA WA Agent for Microsoft SQL Server.
- Due to a Microsoft SQL Server restriction, the agent cannot run multiple MSSQL Server jobs with the same name simultaneously. If a MSSQL Server job is running, you cannot rerun the job until its previous execution finishes.
- MSSQL Server jobs do not produce a spool file.

Define a MSSQL Server Job

You can define a MSSQL Server job to execute predefined jobs in the Microsoft SQL Server Agent to perform administrative tasks.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the MSSQL Server job in the Palette view, and drag the job to the workspace.

   The MSSQL Server icon appears on the Application workspace view.

3. Right-click the MSSQL Server icon, and select Edit from the pop-up menu.

   The Basic page of the MSSQL Server dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent that runs the MSSQL Server job.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Job name/job ID

Specifies the job name or internal job identification (ID) number. Select the Job name or Job ID option button in the MSSQL server job section and then specify the job name or job ID in this field.

Limits: Up to 128 characters for job name; up to 36 characters for job ID

Notes:

- You can type the job name or job ID or, if you are connected to the server, click the arrow search button next to the field to search for a job that is defined in Microsoft SQL Server Agent.
- The agent does not support double quotes in the job name. If the job name includes double quotes, remove the double quotes from the job definition using Microsoft SQL Server Management Studio.

5. (Optional) Specify the following additional information:

Target DB

Specifies the target server database. The agent retrieves the JDBC connection information from the corresponding property file (*target_database*.properties) stored on the agent.

Limits: Up to 128 alphanumeric or underscore characters. The first character cannot be a number.

Notes:

- If the target database is not specified in the job definition, the agent uses the default value specified in the mssql.default.TargetDB parameter in the agentparm.txt file.
- If the target server database is not specified in either the job definition or the mssql.default.TargetDB parameter, the agent reports an error.

MSSQL server name

Specifies the target server to start the job on.

Limits: Up to 128 characters

Note: You can type the MSSQL server name or, if you are connected to the server, click the arrow search button next to the field to search for a target server that the job can run on.

Step name

Specifies the name of the job step to start the job from. The default is to start the job from the first step. To start the job from another step, specify the step name in this field.

Limits: Up to 128 characters

Note: You can type the step name of a job or, if you are connected to the server, click the arrow search button next to the field to search for a job step that is defined in the Microsoft SQL Server Agent job.

User name

Specifies the user that the agent uses to create a JDBC connection to the database. The user can be a Windows domain user for Windows domain authentication or a Microsoft SQL Server user for Microsoft SQL Server authentication. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: Up to 128 alphanumeric characters; you can also specify spaces and special characters including period (.), hyphen (-), and underscore (_).

Examples: Bob, Production:Bob

Note: The drop-down list displays all of the user names that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Domain

Specifies the Windows domain name for Windows authentication.

Limits: Up to 24 alphanumeric characters; you can also specify hyphen (-) and period (.).

Notes:

- If the User name and Domain fields are specified, the agent assumes that the User name contains the Windows domain user. In this case, Windows domain authentication is used.
- If the User name is specified without the Domain field, the agent assumes that the User name contains the Microsoft SQL Server user. In this case, Microsoft SQL Server authentication is used.
- If neither the User name or the Domain fields are specified, the agent uses the default user settings specified in the *target_database*.properties file.

6. Click OK.

   The MSSQL Server job is defined.

Example: Run a Predefined Job on a Target Server

This example runs a predefined job named mssql_job on the myserver target server.

To run a predefined job on a target server

1. Enter the following information on the Basic page of the job definition:
   - Name—mssql_DE_job
   - Agent name—AGENT_MSSQL
   - MSSQL server name—myserver
   - Target DB—mytargetDB
   - User name—mydomainuser
   - Domain—mydomain
2. Select the Job name option button and enter **mssql_job** in the Job name/job ID field.
3. Click OK.

More information:

MSSQL Server Jobs (see page 308)

Search for a Microsoft SQL Server Agent Job

You can use the Search MSSQL Jobs dialog to search for and select a job that is defined in Microsoft SQL Server Agent.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the MSSQL Server job definition.
3. Specify the name of the agent that runs the MSSQL Server job in the Agent name field.
4. Click the arrow search button next to the Job name/job ID field.

   The Search MSSQL Jobs dialog opens.

   Note: The fields in this dialog are populated with the information that is specified on the Basic page of the job definition, if any.
5. (Optional) Specify the filter criteria in the following fields in the Filter Options section:

   Job Type

   Specifies the job type to filter on. Options are LOCAL and MULTI-SERVER. The default is to return all jobs regardless of job type.

   Owner Login Name

   Specifies the login name of the owner of the job to filter on. The default is to return all jobs regardless of the owner login name.

   Category

   Specifies the category name to filter on. Options are as follows:

   - Uncategorized (Local)
   - Data Collector
   - Database Engine Tuning Advisor
   - Database Maintenance
   - Full-Text
   - Jobs From MSX
   - Log Shipping
   - REPL-Alert Response
   - REPL-Checkup
   - REPL-Distribution
   - REPL-Distribution Cleanup
   - REPL-History Cleanup

- REPL-LogReader
- REPL-Merge
- REPL-QueueReader
- REPL-Snapshot
- REPL-Subscription Cleanup

The default is to return all jobs regardless of category.

Subsystem

Specifies the name of the subsystem to filter on. Options are as follows:

- TSQL
- ActiveScripting
- CmdExec
- Snapshot
- LogReader
- Distribution
- Merge
- QueueReader
- AnalysisQuery
- AnalysisCommand
- SSIS
- PowerShell

Note: When this field is specified, only jobs with job steps that belong to the specified subsystem are listed. The default is to return all jobs regardless of subsystem.

Execution Status

Specifies the execution status to filter on. Options are as follows:

Suspended

Lists only jobs that are not idle or suspended.

Executing

Lists only jobs that are executing.

WaitingForThread

Lists only jobs that are waiting for a thread.

BetweenRetries

Lists only jobs that are between retries.

Idle

Lists only jobs that are idle.

PerformingCompletionActions

Lists only jobs that are performing completion options.

The default is to return all jobs regardless of execution status.

Date

Specifies a date range based on when the job was created or last modified. To filter by date, specify the following:

- Type of date (DateCreated or DateLastModified)
- Date comparator (After or Before)
- Date (using the calendar)
- Time

Example: DateCreated After Sept 13, 2013 12:00:00 AM (returns jobs that were created since midnight on September 13, 2013).

Enabled

Indicates whether to list only enabled or disabled jobs.

TRUE

Lists only enabled jobs.

FALSE

Lists only disabled jobs.

The default is to return enabled and disabled jobs.

Description

Specifies the job description to filter on. The Description field can include the wildcard characters *, ?, ^, and [] for pattern matching. The * wildcard character is equivalent to the % SQL wildcard character. The ? wildcard character is equivalent to the _ SQL wildcard character. The default is to return all jobs regardless of description.

6. Click the arrow search button next to the MSSQL server info section.

   The List of Jobs table is populated with the list of jobs that meet the search criteria.

   Notes:

   - You can sort the data in the table by clicking on a table column heading.
   - The agent administrator can configure the properties files on the agent to change the fields that are displayed in the table. For more information about configuring the properties files and the available fields that can be displayed, see the *CA Workload Automation Agent for Microsoft SQL Server Implementation Guide*.

7. Select the job you want from the table.
8. Click Select Job Name to specify the job name or click Select Job ID to specify the job ID in the job definition.

   The specified job name or job ID of the selected job is set on the Basic page of the job definition.

Example: Retrieve a List of Microsoft SQL Server Agent Jobs Based on Job Description Criteria

Suppose that you want to retrieve a list of all jobs with the job description that starts with sqljob.

To list the Microsoft SQL Server Agent jobs based on job description criteria

1. Open the Basic page of the MSSQL Server job definition.
2. Enter **AGENT_MSSQL** in the Agent name field.
3. Click the arrow search button next to the Job name/job ID field.

   The Search MSSQL Jobs dialog opens.
4. Enter **sqljob*** in the Description field.
5. Click the arrow search button next to the MSSQL server info section.

   The List of Jobs table is populated with the list of jobs that meet the search criteria.

Search for a Target Server that the Microsoft SQL Server Agent Job Can Run on

You can use the Search MSSQL Job Target Server dialog to search for and select a server that the Microsoft SQL Server Agent job can run on.

Note: To list the target servers that the job can run on, you must enlist the target servers into the master server in the Microsoft SQL Server database.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the MSSQL Server job definition.
3. Complete the following required fields:

   Agent name

   Specifies the name of the agent that runs the MSSQL Server job.

   Job name/job ID

   Specifies the job name or internal job identification (ID) number. Select the Job name or Job ID option button in the MSSQL server job section and then specify the job name or job ID in this field.
4. Click the arrow search button next to the MSSQL server name field.

   The Search MSSQL Job Target Servers dialog displays the list of target servers that the job can run on.

   Notes:

   - You can sort the data in the List of Target Servers table by clicking on a table column heading.
   - The agent administrator can configure the properties files on the agent to change the fields that are displayed in the table. For more information about configuring the properties files and the available fields that can be displayed, see the *CA Workload Automation Agent for Microsoft SQL Server Implementation Guide*.
5. Select the server that you want to run the job on from the table.
6. Click Select Job Target Server Name to specify the target server in the job definition.

   The selected target server is set on the Basic page of the job definition.

Example: Retrieve a List of Target Servers that the Microsoft SQL Server Agent Job Can Run on

Suppose that you want to retrieve a list of all target servers that the job named Test job monitoring can run on.

To list the target servers that the Microsoft SQL Server Agent job can run on

1. Open the Basic page of the MSSQL Server job definition.
2. Enter **AGENT_MSSQL** in the Agent name field.
3. Select the Job name option button and enter **Test job monitoring** in the Job name/job ID field.
4. Click the arrow search button next to the MSSQL server name field.

   The Search MSSQL Job Target Servers dialog displays the list of target servers that the job can run on.

Search for a Job Step in a Microsoft SQL Server Agent Job

You can use the Search MSSQL Job Steps dialog to search for and select a job step that is defined in the Microsoft SQL Server Agent job. You can start a job from the job step that you want.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the MSSQL Server job definition.
3. Complete the following required fields:

   Agent name

   Specifies the name of the agent that runs the MSSQL Server job.

   Job name/job ID

   Specifies the job name or internal job identification (ID) number. Select the Job name or Job ID option button in the MSSQL server job section and then specify the job name or job ID in this field.
4. Click the arrow search button next to the Step name field.

   The Search MSSQL Job Steps dialog opens.
5. (Optional) Specify the search criteria in *one* of the following fields:

   Job Step Name

   Specifies the name of the job step to filter on.

   Job Step ID

   Specifies the ID of the job step to filter on.

   The default is to include all job steps for the specified field.
6. Click the arrow search button next to the MSSQL Job Step Filter Info section.

   The List of Job Steps table is populated with the job steps that meet the search criteria for the specified job.

   Notes:

   - You can sort the data in the table by clicking on a table column heading.
   - The agent administrator can configure the properties files on the agent to change the fields that are displayed in the table. For more information about configuring the properties files and the available fields that can be displayed, see the *CA Workload Automation Agent for Microsoft SQL Server Implementation Guide*.

7. Select the job step that you want to start the job from in the table.

   The default is to start the job from the first step.

8. Click Select Job Step Name to specify the step name in the job definition.

   The specified step name of the selected job step is set on the Basic page of the job definition.

Example: Display Details of All Job Steps

Suppose that you want to display details of all job steps for a job named Test job monitoring.

To display details of all job steps

1. Open the Basic page of the MSSQL Server job definition.
2. Enter **AGENT_MSSQL** in the Agent name field.
3. Select the Job name option button and enter **Test job monitoring** in the Job name/job ID field.
4. Click the arrow search button next to the Step name field.

   The Search MSSQL Job Steps dialog opens.

5. Click the arrow search button next to the MSSQL Job Step Filter Info section.

   The List of Job Steps table is populated with the details of all the job steps that are defined in the job.

Example: Display Details of a Job Step

Suppose that you want to display details of a job step named Step 2 for a job named Test job monitoring.

To display details of a job step

1. Open the Basic page of the MSSQL Server job definition.
2. Enter **AGENT_MSSQL** in the Agent name field.
3. Select the Job name option button and enter **Test job monitoring** in the Job name/job ID field.
4. Click the arrow search button next to the Step name field.

   The Search MSSQL Job Steps dialog opens.

5. Enter **Step 2** in the Job Step Name field.
6. Click the arrow search button next to the MSSQL Job Step Filter Info section.

   The List of Job Steps table is populated with the details of only Step 2.

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:

Copy Single Request

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

Request Set

Runs multiple programs in an Oracle Applications application.

Single Request

Runs a single program in an Oracle Applications application.

Define a Copy Single Request Job

You can define a Copy Single Request job to copy an existing single request defined on Oracle Applications and run it under the agent. To copy an existing single request, you must provide its request ID in the job definition. 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Copy Single Request job from the Oracle E-Business Suite group in the Palette view, and drag the job to the workspace.

   The Copy Single Request icon appears on the Application workspace view.

3. Right-click the Copy Single Request job icon, and select Edit from the pop-up menu.

   The Basic page of the Copy Single Request dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

Original 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

5. (Optional) Specify the following additional information to override values in the existing single request:

User

Specifies the Oracle Applications user name that the job runs under. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.user parameter in the agent's agentparm.txt file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

Responsibility

Specifies an Oracle Applications responsibility name.

Note: Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.responsibility parameter in the agent's agentparm.txt file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

Note: To override the values specified in the existing single request, both the user name and responsibility name are required. If the user name or responsibility name is *not* specified (in either the job definition or as a default in the agentparm.txt file), the job copies the user name and responsibility name from the original single request.

Monitor children

Indicates whether the agent monitors the children of Oracle Applications programs. Program children are programs that are released by the parent program. Up to five levels of children can be monitored for each job.

Notes:

- You must select Yes in the Monitor children drop-down list to enable the Monitor children delay field.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.monChildren parameter in the agent's agentparm.txt file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

Monitor children delay

Specifies the number of milliseconds to wait after a parent completes before monitoring children. When monitoring children, use this field to ensure the agent receives accurate information about the status of program children.

Limits: Up to 5 digits

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

6. Click OK.

   The Copy Single Request job is defined.

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 Oracle Applications user name and responsibility name. The children of the single request program are monitored 60 seconds after the program completes.

To copy a single request

1. Enter the following information in the Basic page:
   - Name—COPYJOB
   - Agent name—CYB0A
   - Original request ID—2255470
   - User—SYSADMIN
   - Responsibility—System Administrator
2. Select Yes from the Monitor children drop-down list.
3. Enter **60000** in the Monitor children delay field.
4. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

Define a Request Set Job

You can define a Request Set 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 Oracle Applications request set:

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Request Set job from the Oracle E-Business Suite group in the Palette view, and drag the job to the workspace.

   The Request Set icon appears on the Application workspace view.

3. Right-click the Request Set job, and select Edit from the pop-up menu

   The Basic page of the Request Set dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   OA Application

   Indicates whether the Name field in the OA Application section contains the display name or short name of the Oracle Applications application. Options are Display name and Short name.

   Default: Display name

Name (OA Application section)

Specifies the name of the Oracle Applications application. You can identify the application by its display name or its short name. In Oracle Applications, the display name is part of the request definition and is found in the Application field. The short name is defined by the Oracle Applications Concurrent Manager.

Note: You can type the display or short name or, if you are connected to the server, click the arrow browse button next to the field to search for an application defined in Oracle Applications.

Request set

Indicates whether the Name field in the Request set section contains the display name or short name of the Oracle Applications request set. Options are Display name and Short name.

Default: Display name

Name (Request set section)

Specifies the name of the Oracle Applications request set. You can identify the request set by its display name or its short name. 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 240 characters

Note: You can type the display or short name or, if you are connected to the server, click the arrow browse button next to the field to search for a request set defined in Oracle Applications.

5. Click Other OA Parameters in the left pane.

   The Other OA Parameters page opens in the right pane.

6. Specify the following additional information as appropriate:

   User name

   Specifies the Oracle Applications user name that the job runs under.

   Examples: APPSMGR, SYSADMIN

   Notes:

   - If you do not specify this field, 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 User name field overrides the default user name specified in the agent's agentparm.txt.

Responsibility

Specifies an Oracle Applications responsibility name.

Notes:

- If you do not specify this field, 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 Responsibility field overrides the default responsibility name specified in the agent's agentparm.txt.

7. (Optional) Specify the following additional information:

Use argument defaults

Indicates whether to use Oracle Applications argument defaults for arguments not defined in this page.

Notes:

- To resolve a default value, the agent can use a constant value, look up a variable in a user's profile, run an SQL statement, or convert a date to the appropriate format.
- You can also specify value set expressions (such as flexfields) for default values in the Default Argument Resolution page of the job definition.
- The agent cannot resolve the segment or field default types.
- If a default value is not specified for a required date parameter, the agent uses SYSDATE (system date) as the default value.
- You must select Yes in the Use argument defaults drop-down list to enable the "Put default values in quotes" and "Use request set defaults first" fields.

Put default values in quotes

Indicates whether to quote resolved expressions in default values.

Note: This option does not apply to expressions defined in the Default Argument Resolution page of the job definition.

Use request set defaults first

Indicates whether request set defaults take precedence over program defaults.

Default: If both request set defaults and program defaults are specified, the agent uses the program defaults.

Note: Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.useSetDefaultsFirst parameter in the agent's agentparm.txt file. The value you specify in the job definition overrides the default value in the agentparm.txt file.

Monitor children

Indicates whether the agent monitors the children of Oracle Applications programs. Program children are programs that are released by the parent program. Up to five levels of children can be monitored for each job.

Notes:

- You must select Yes in the Monitor children drop-down list to enable the Monitor children delay field.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.monChildren parameter in the agent's agentparm.txt file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

Monitor children delay

Specifies the number of milliseconds to wait after a parent completes before monitoring children. When monitoring children, use this field to ensure the agent receives accurate information about the status of program children.

Limits: Up to 5 digits

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

Save output

Indicates whether to save the output from an Oracle Applications single request or request set.

8. (Optional) Expand the Layout section and specify the following additional information:

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 10 characters

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

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 10 characters

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

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 10 characters

Note: You must specify a value for this field to enable the "Template language" and "Template territory" fields.

9. (Optional) Expand the Print the output to section and specify the following additional information:

Printer name

Specifies the Oracle Applications printer name.

Print style

Specifies the Oracle Applications print style.

Print copies

Specifies the number of copies to print.

10. (Optional) Click Add to specify the following program data for the first program in the New Program Data dialog:

Program sequence number

Specifies the program's Oracle Applications order number.

Program arguments

Defines the argument values to override Oracle Applications default values.

Note: Separate argument values with commas. Do not add spaces.

Put default values in quotes

Indicates whether to quote resolved expressions in default values.

Note: This option does not apply to expressions defined in the Default Argument Resolution page of the job definition.

Save output

Indicates whether to save the output from an Oracle Applications single request or request set.

Template language

Specifies the template language for a program in a request set.

Template territory

Specifies the template territory for a program in a request set.

Format

Specifies the output format for a program in a request set.

Note: You must specify a value for this field to enable the "Template language" and "Template territory" fields.

Short name

Specifies a list of short user names to notify when a program in a request set completes. Click Add to add individual user names to the Short names table.

Limits: Up to 1024 characters

Note: You can type each short name or, if you are connected to the server, click Get to search for user names defined in Oracle Applications.

Display name

Specifies a list of display user names to notify when a program in a request set completes. Click Add to add individual user names to the Display names table.

Limits: Up to 1024 characters

Note: You can type each display name or, if you are connected to the server, click Get to search for user names defined in Oracle Applications.

Printer name

Specifies the Oracle Applications printer name.

Print style

Specifies the Oracle Applications print style.

Print copies

Specifies the number of copies to print.

11. Click OK.

    The Program Data dialog closes and the program data is added to the Program Data table.

12. (Optional) Repeat the previous two steps to specify program data for additional programs.

13. (Optional) Click Default Argument Resolution in the left pane to specify value set expressions for default values.

    The Default Argument Resolution page opens in the right pane.

14. Click Add and specify the following information in the first row of the table:

    Expression

    Specifies the expression to resolve. 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 in the table.

    Limits: Up to 256 characters

    Examples:

    - :$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT
    - :$PROFILES$.GL_ACCESS_SET_ID

    Default 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

    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.

15. (Optional) Repeat the previous step to add more default value expressions to the table.

16. Click OK.

    The Request Set job is defined.

Example: Run Two Programs with Different Arguments and Printing Options

Suppose that you want to define a Request Set job named AOL405.AP035 to run multiple programs on the CYBOA agent. The request set EXTRACTS belongs to the Application Object Library application in Oracle E-Business Suite and runs two programs with different arguments and printing options. The job runs under the SYSADMIN user with System Administrator responsibility.

To run two programs with different arguments and printing options

1. Enter the following information in the Basic page:
   - Name—AOL405
   - Qualifier—AP035
   - Agent name—CYBOA
2. Select the Display name option button in the OA Application section and enter **Application Object Library** in the Name (OA Application) field.
3. Select the Short name option button in the Request set section and enter **EXTRACTS** in the Name (Request set) field.
4. Enter the following information in the Other OA Parameters page:
   - User name—SYSADMIN
   - Responsibility—System Administrator
5. Click Add and complete the following fields for the first program in the Program Data dialog:
   - Program sequence number—1
   - Program arguments—,12345,2,,3223,2,,1
   - Printer name—Q_DEVELOPMENT
   - Print style—Portrait
   - Print copies—5
6. Select Yes from the Save output drop-down list.
7. Click OK.

   The program data for the first program is added to the Program Data table.

8. Click Add and complete the following fields for the second program in the Program Data dialog:
   - Program sequence number—2
   - Program arguments—12345,,,,1,1755,,-1,,
   - Printer name—Q_8
   - Print style—Portrait
   - Print copies—8
9. Click OK.

   The program data for the second program is added to the Program Data table.
10. Click OK.

Example: Run Multiple Programs with Different Arguments and Notification Users

Suppose that you want to define a Request Set job named AOL405.AP036 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. The job runs under the SYSADMIN user with System Administrator responsibility.

To run multiple programs with different arguments and notification users

1. Enter the following information in the Basic page:
   - Name—AOL405
   - Qualifier—AP036
   - Agent name—CYBOA
2. Select the Display name option button in the OA Application section and enter **Application Object Library** in the Name (OA Application) field.
3. Select the Short name option button in the Request set section and enter **FNDCPRT_SET** in the Name (Request set) field.
4. Enter the following information in the Other OA Parameters page:
   - User name—SYSADMIN
   - Responsibility—System Administrator
5. Select Yes in the Use arguments defaults drop-down list.
6. Select Yes in the Put default values in quotes drop-down list.

7. Specify the following information for the first program:
   a. Click Add and complete the following fields in the Program Data dialog:
      - Program sequence number—1
      - Program arguments—,,,X,
   b. Enter **System Administrator** in the Display name field in the Notify the following people section, and click Add.

      The user is added to the Display names table.
   c. Enter **Admission Administrator** in the Display name field in the Notify the following people section, and click Add.

      The user is added to the Display names table.
   d. Click OK.

   The program data for the first program is added to the Program Data table.
8. Specify the following information for the second program:
   a. Click Add and complete the following fields in the Program Data dialog:
      - Program sequence number—2
      - Program arguments—,2,,4
   b. Enter **ASGUEST** in the Short name field in the Notify the following people section, and click Add.

      The user is added to the Short names table.
   c. Click OK.

   The program data for the second program is added to the Program Data table.
9. Specify the following information for the seventh program:
   a. Click Add and complete the following fields in the Program Data dialog:
      - Program sequence number—7
      - Program arguments—,,arg3
   b. Click OK.

   The program data for the seventh program is added to the Program Data table.
10. Click OK.

More information:

Search for an Oracle Applications Application (see page 349)
Search for an Oracle Applications Request Set (see page 334)
Search for Users to Notify Upon Completion of a Program in a Request Set (see page 335)

Search for an Oracle Applications Request Set

You can use the List Request Sets dialog to search for and select a request set defined in Oracle Applications.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the Oracle E-Business Suite Request Set job definition.
3. Click the arrow browse button next to the Name field in the Request set section.

   The List Request Sets dialog opens.
4. Complete the following required field:

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   Default: Agent name specified on the Basic page of the job definition
5. (Optional) Specify the search criteria in the following fields:

   Short name

   Specifies the short name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 50 characters

   Display name

   Specifies the display name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 240 characters
6. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of request sets that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.
7. Select the request set you want from the Results table.
8. Click Select Short Name to specify the short name or click Select Display Name to specify the display name in the job definition.

   The specified short name or display name of the selected request set is set in the Basic page of the job definition.

Example: Retrieve a List of Request Sets Based on Display Name Criteria

Suppose that you want to retrieve a list of all request sets with display names that start with A.

To retrieve a list of request sets based on display name criteria

1. Open the Basic page of the Oracle E-Business Suite Request Set job definition.
2. Click the arrow browse button next to the Name field in the Request set section.

   The List Request Sets dialog opens.
3. Enter the following information:
   - Agent name—CYBOA
   - Display name—A*
4. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of request sets that meet the search criteria.

Search for Users to Notify Upon Completion of a Program in a Request Set

You can use the List Users dialog to search for and select users defined in Oracle Applications to notify when a program in a request set completes. You can select users by short name or display name, or both.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Other OA Parameters page of the Oracle E-Business Suite Request Set job definition.
3. Click Add to specify program data for a program in a request set.

   The New Program Data dialog opens.
4. Do *one* of the following:
   - To select users by short name, click the Get button next to the Short names table.
   - To select users by display name, click the Get button next to the Display names table.

   The List Users dialog opens.
5. Complete the following required field:

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   Default: Agent name specified on the Basic page of the job definition

6. (Optional) Specify the search criteria in the following fields:

   Short name

   Specifies the short name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 50 characters

   Display name

   Specifies the display name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 240 characters

7. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of users that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.

8. Select the users you want from the Results table.
9. Click Select Short Name/Select Display Name.

   The selected user names appear in the Short names or Display names table in the New Program Data dialog.

10. Click OK.

    The New Program Data dialog closes.

Define a Single Request Job

You can define a Single Request 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 Oracle Applications single request:

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Single Request job from the Oracle E-Business Suite group in the Palette view, and drag the job to the workspace.

   The Single Request icon appears on the Application workspace view.

3. Right-click the Single Request job icon, and select Edit from the pop-up menu

   The Basic page of the Single Request dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   OA Application

   Indicates whether the Name field in the OA Application section contains the display name or short name of the Oracle Applications application. Options are Display name and Short name.

   Default: Display name

Name (OA Application section)

Specifies the name of the Oracle Applications application. You can identify the application by its display name or its short name. In Oracle Applications, the display name is part of the request definition and is found in the Application field. The short name is defined by the Oracle Applications Concurrent Manager.

Note: You can type the display or short name or, if you are connected to the server, click the arrow browse button next to the field to search for an application defined in Oracle Applications.

Program

Indicates whether the Name field in the Program section contains the display name or the short name of the program. Options are Display name and Short name.

Default: Display name

Name (Program section)

Specifies the name of the Oracle Applications single request program. You can identify the program by its display name or its short name. 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 240 characters

Note: You can type the display or short name or, if you are connected to the server, click the arrow next to the field to search for a program defined in Oracle Applications.

5. Click Other OA Parameters in the left pane.

   The Other OA Parameters page opens in the right pane.

6. Specify the following additional information as appropriate:

   User name

   Specifies the Oracle Applications user name that the job runs under.

   Examples: APPSMGR, SYSADMIN

   Notes:

   - If you do not specify this field, 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 User name field overrides the default user name specified in the agent's agentparm.txt.

Responsibility

Specifies an Oracle Applications responsibility name.

Notes:

- If you do not specify this field, 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 Responsibility field overrides the default responsibility name specified in the agent's agentparm.txt.

7. (Optional) Specify the following additional information:

Description

Specifies the Oracle Applications description.

Program arguments

Defines the argument values to override Oracle Applications default values.

Note: Separate argument values with commas. Do not add spaces.

Use argument defaults

Indicates whether to use Oracle Applications argument defaults for arguments not defined in this page.

Notes:

- To resolve a default value, the agent can use a constant value, look up a variable in a user's profile, run an SQL statement, or convert a date to the appropriate format.
- You can also specify value set expressions (such as flexfields) for default values in the Default Argument Resolution page of the job definition.
- The agent cannot resolve the segment or field default types.
- If a default value is not specified for a required date parameter, the agent uses SYSDATE (system date) as the default value.
- You must select Yes in the Use argument defaults drop-down list to enable the Put default values in quotes field.

Put default value in quotes

Indicates whether to quote resolved expressions in default values.

Note: This option does not apply to expressions defined in the Default Argument Resolution page of the job definition.

Monitor children

Indicates whether the agent monitors the children of Oracle Applications programs. Program children are programs that are released by the parent program. Up to five levels of children can be monitored for each job.

Notes:

- You must select Yes in the Monitor children drop-down list to enable the Monitor children delay field.
- Your agent administrator can specify a default setting for all Oracle E-Business Suite jobs by setting the oa.default.monChildren parameter in the agent's agentparm.txt file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

Monitor children delay

Specifies the number of milliseconds to wait after a parent completes before monitoring children. When monitoring children, use this field to ensure the agent receives accurate information about the status of program children.

Limits: Up to 5 digits

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

Save output

Indicates whether to save the output from an Oracle Applications single request or request set.

8. (Optional) Expand the Layout section and specify the following additional information:

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 10 characters

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

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 10 characters

Note: 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 value you enter in the job definition overrides the default value in the agentparm.txt file.

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 10 characters

Note: You must specify a value for this field to enable the "Template language" and "Template territory" fields.

9. (Optional) Expand the Notify the following people section and specify the following additional information:

Short name

Specifies a list of short user names to notify when the single request completes. Click Add to add individual user names to the Short names table.

Limits: Up to 1024 characters

Note: You can type each short name or, if you are connected to the server, click Get to browse for user names defined in Oracle Applications.

Display name

Specifies a list of display user names to notify when the single request completes. Click Add to add individual user names to the Display names table.

Limits: Up to 1024 characters

Note: You can type each display name or, if you are connected to the server, click Get to browse for user names defined in Oracle Applications.

10. (Optional) Expand the Print the output to section and specify the following additional information:

Printer name

Specifies the Oracle Applications printer name.

Print style

Specifies the Oracle Applications print style.

Print copies

Specifies the number of copies to print.

11. (Optional) Click Default Argument Resolution in the left pane to specify value set expressions for default values.

    The Default Argument Resolution page opens in the right pane.

12. Click Add and specify the following information in the first row of the table:

    Expression

    Specifies the expression to resolve. 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 in the table.

    Limits: Up to 256 characters

    Examples:

    - :$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT
    - :$PROFILES$.GL_ACCESS_SET_ID

    Default 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

    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.

13. (Optional) Repeat the previous step to add more default value expressions to the table.

14. Click OK.

    The Single Request job is defined.

Example: Run a Single Request Program

Suppose that you want to define a Single Request job named AOL1805.AP034 to run a single request program on the CYBOA agent. The single request belongs to the Application Object Library application in Oracle Applications and runs the program FNDSCARU. The job runs under the SYSADMIN user with System Administrator responsibility.

To run a Single Request program

1. Enter the following information in the Basic page:
   - Name—AOL1805
   - Qualifier—AP034
   - Agent name—CYBOA
2. Select the Display name option button in the OA Application section and enter **Application Object Library** in the Name (OA Application) field.
3. Select the Short name option button in the Program section and enter **FNDSCARU** in the Name (Program) field.
4. Enter the following information in the Other OA Parameters page:
   - User name—SYSADMIN
   - Responsibility—System Administrator
5. Click OK.

Example: Pass Oracle E-Business Suite Argument Values

Suppose that you want to pass the following argument values to a single request program, assuming that the second and fifth arguments have default values defined in Oracle Applications:

T,*DefArg2*,X23,,*DefArg5*

Since no default is specified for the fourth argument, the agent passes an empty string.

To pass Oracle E-Business Suite argument values

1. Open the OA Single Request job for which you want to pass Oracle Applications argument values.
2. Click Other OA Parameters in the left pane.

   The Other OA Parameters page opens in the right pane.
3. Enter **T,,X23,,** in the Program arguments field.
4. Select Yes from the Use argument defaults drop-down list.
5. Click OK.

Example: Specify Layout Template Options

Suppose that you want to define a Single Request job to run a single request program named AOL1805.AP035 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.

To specify layout template options

1. Enter the following information in the Basic page:
   - Name—AOL1805
   - Qualifier—AP035
   - Agent name—CYBOA
2. Select the Short name option button in the OA Application section and enter **FND** in the Name (OA Application) field.
3. Select the Short name option button in the Program section and enter **FNDSCURS** in the Name (Program) field.
4. Enter the following information in the Other OA Parameters page:
   - User name—SYSADMIN
   - Responsibility—System Administrator
   - Format—PDF
   - Template language—EN
   - Template territory—US
5. Click OK.

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'
```

To specify a default value set expression

1. Enter the following information in the Basic page:
   - Name—AOL1805
   - Qualifier—AP036
   - Agent name—CYBOA
2. Select the Display name option button in the OA Application section and enter **Process Manufacturing Financials** in the Name (OA Application) field.
3. Select the Short name option button in the Program section and enter **GMFDSUR** in the Name (Program) field.
4. Enter the following information in the Other OA Parameters page:
   - User name—SYSADMIN
   - Responsibility—System Administrator
5. Select Yes in the Use arguments defaults drop-down list.

6. Click Add in the Default Argument Resolution page.

   A row is added to the Default values expressions table.

7. Enter **:$FLEX$.EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT** in the Expression column.

8. Enter '**SELECT EAM_SRS_MAINTAINED_ITEM_NAMES_W_ACT from SOME_TABLE where SOME_KEY = 'Default''** in the Default Value column.

9. Click OK.

Search for an Oracle Applications Single Request Program

You can use the List Programs dialog to search for and select a single request program defined in Oracle Applications.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the Oracle E-Business Suite Single Request job definition.
3. Click the arrow browse button next to the Name field in the Program section.

   The List Programs dialog opens.

4. Complete the following required field:

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   Default: Agent name specified on the Basic page of the job definition

5. (Optional) Specify the search criteria in the following fields:

   Short name

   Specifies the short name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 50 characters

   Note: This field is populated with the short name specified in the Basic page of the job definition, if any.

   Display name

   Specifies the display name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 240 characters

   Note: This field is populated with the display name specified in the Basic page of the job definition, if any.

6. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of programs that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.

7. Select the program you want from the Results table.
8. Click Select Short Name to specify the short name or click Select Display Name to specify the display name in the job definition.

   The specified short name or display name of the selected program is set in the Basic page of the job definition.

Example: Retrieve a List of Programs Based on Short Name and Display Name Criteria

Suppose that you want to retrieve a list of all programs with short names that start with AZR12 and display names that start with A.

To retrieve a list of programs based on short name and display name criteria

1. Open the Basic page of the Oracle E-Business Suite Single Request job definition.
2. Click the arrow browse button next to the Name field in the Program section.

   The List Programs dialog opens.

3. Enter the following information:
   - Agent name—AGENT
   - Short name—AZR12*
   - Display name—A*
4. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of programs that meet the search criteria.

Search for Users to Notify Upon Completion of the Single Request

You can use the List Users dialog to search for and select users defined in Oracle Applications to notify when the single request completes. You can select users by short name or display name, or both.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Other OA Parameters page of the Oracle E-Business Suite Single Request job definition.
3. Do *one* of the following:
   - To select users by short name, click the Get button next to the Short names table.
   - To select users by display name, click the Get button next to the Display names table.

   The List Users dialog opens.
4. Complete the following required field:

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   Default: Agent name specified on the Basic page of the job definition
5. (Optional) Specify the search criteria in the following fields:

   Short name

   Specifies the short name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 50 characters

   Display name

   Specifies the display name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 240 characters

6. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of users that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.

7. Select the users you want from the Results table.
8. Click Select Short Name/Select Display Name.

   The selected user names appear in the Short names or Display names table in the Other OA Parameters page.

Search for an Oracle Applications Application

Note: This procedure applies to the Oracle E-Business Suite Single Request and Request set jobs.

You can use the List Applications dialog to search for and select an application defined in Oracle Applications.

Follow these steps:

1. Ensure you are connected to the server.
2. Open the Basic page of the Oracle E-Business Suite Single Request or Request Set job definition.
3. Click the arrow browse button next to the Name field in the OA Application section.

   The List Applications dialog opens.

4. Complete the following required field:

   Agent name

   Specifies the name of the Oracle E-Business Suite agent where the job runs.

   Default: Agent name specified on the Basic page of the job definition

5. (Optional) Specify the search criteria in the following fields:

   Short name

   Specifies the short name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 50 characters

   Display name

   Specifies the display name filter. You can use wildcard characters asterisk (*) to represent zero or more characters and question mark (?) to represent a single character.

   Limits: Up to 240 characters

6. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of applications that meet the search criteria.

   Note: You can sort the data in the table by clicking on a table column heading.

7. Select the application you want from the Results table.
8. Click Select Short Name to specify the short name or click Select Display Name to specify the display name in the job definition.

   The specified short name or display name of the selected application is set in the Basic page of the job definition.

Example: Retrieve a List of Applications Based on Short Name Criteria

Suppose that you want to retrieve a list of all applications with short names that start with A.

To retrieve a list of Applications based on short name criteria

1. Open the Basic page of the Oracle E-Business Suite Single Request or Request Set job definition.
2. Click the arrow browse button next to the Name field in the OA Application section.

   The List Applications dialog opens.

3. Enter the following information:
   - Agent name—AGENT
   - Short name—A*
4. Click the arrow search button next to the Search criteria section.

   The Results table is populated with the list of applications that meet the search criteria.

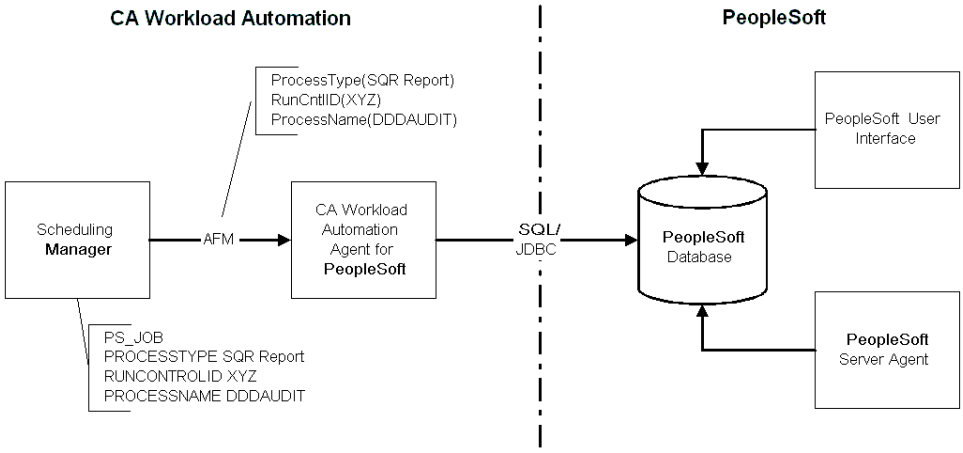
PeopleSoft Jobs

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:



Define a PeopleSoft Job to Run a Request

You can define a PeopleSoft 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.

Do not use the following special characters when defining PeopleSoft jobs: left parenthesis (( ) and right parenthesis ( )) and apostrophe ('). Use caution when using other special characters, such as backslash (\) and at (@).

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the PeopleSoft job from the Palette view, and drag the job to the workspace.

   The PeopleSoft icon appears on the Application workspace view.

3. Right-click the PeopleSoft icon, and select Edit from the pop-up menu.

   The Basic page of the PeopleSoft dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that runs the PeopleSoft request.

   Process name

   Specifies the name of the PeopleSoft process you want the job to run. PeopleSoft stores the list of process names in the PS_PRCSDEFN table.

   Example: AEMINITEST

Process type

Specifies the PeopleSoft process type corresponding to the process that you want to run. PeopleSoft stores the list of process types in the PS_PRCSTYPEDEFN table. This value corresponds to the Process Type field in PeopleSoft.

Example: Application Engine

Note: If the process type corresponding to the process that you want to run does not appear in the drop-down list, you can type the process type in the field.

5. Click Other PS Parameters in the left pane to enter the required PS operator ID and run control ID.

   The Other PS Parameters page opens in the right pane.

6. Complete the following required information, as appropriate:

   PS operator ID

   Specifies the PeopleSoft operator ID under whose authority the report runs. The operator ID must be defined as a user in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Note: This field is mandatory unless its value is specified in the agent parameter (agentparm.txt) file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

   Example: Bob, Production:Bob

   Run control ID

   Identifies a set of PeopleSoft run parameters for a given PeopleSoft process.

   Example: FLOOR8_COLOR

   Note: This field is mandatory unless its value is specified in the agent parameter (agentparm.txt) file. The value you enter in the job definition overrides the default value in the agentparm.txt file.

7. (Optional) Specify the following additional information:

   Server name

   (Optional) Specifies the name of the target server executing the PeopleSoft job. PeopleSoft stores the list of server names in the PS_SERVERDEFN table.

   Example: PSNT

Output destination type

(Optional) Specifies the type of output for the report: NONE, FILE, PRINTER, EMAIL, or WEB.

Note: If you select EMAIL or WEB as the Output destination type, you can distribute the PeopleSoft report electronically to operators, groups of people, or individuals. Use the Distribution page to specify your distribution.

Example: WEB

Output destination format

(Optional) Specifies the type of format for the report output. PeopleSoft stores the list of output destination formats in the PSXLATITEM table.

Example: CSV

Output destination path

(Optional) Defines the path to the output directory. Enter a path to change the output location for FILE and PRINTER output destination types. The path you define in this field overrides the default path defined by the PeopleSoft system.

Example: \\CYBERMATION\Q-SUPPORT

Time zone

(Optional) Specifies a different time zone for the report being run.

Example: Central Time (US)

Arguments

(Optional) Defines an argument string of positional parameters to be appended to the PeopleSoft database. Additional arguments that are specified in this field are appended in a parameter list.

Most parameters for a particular PeopleSoft job are specified in a parameter list in two database tables:

- PS_PRCSTYPEDEFN-Template for a process type
- PS_PRCSDEFN-Additional parameters for a particular process name

Example: -ORIENTL "SHARE"

Skip parameter updates

(Optional) Indicates whether you want the Agent to update job parameters with data in the PS_PRCSDEFN table. If you select Yes, the Agent does not update the parameters in the table.

Note: We recommend you skip parameter updates when some bind variables in the table PS_PRCSDEFN may not be suitably defined. You can use the Arguments field to pass additional argument values.

Disable restart

(Optional) Indicates whether to disable a restart feature for previously failed jobs from the point where the job failed. By default, when a previously-failed job is resubmitted, it will restart from where it was stopped.

8. (Optional) Click Run Control Table List in the left pane to modify run control parameters.

   The Run Control Table List page opens in the right pane.

9. Click Add.

   A new row is added to the Run control table list table.

10. Complete the following fields as required:

    Run control table name

    (Optional) Specifies the table name that contains the run parameters for a given PeopleSoft process.

    Example: PS_AEREQUESTTBL

    Run control arguments

    (Optional) Specifies the run parameters for a PeopleSoft process. Enter a list of argument values separated by commas. You can include symbolic variables in your argument string.

    Note: You must surround arguments containing special characters with double quotation marks.

11. (Optional) Repeat the previous two steps to specify run control parameters in additional run control tables.

12. Click OK.

    The PeopleSoft job is defined.

Example: Add a Row of Process Parameters to a Run Control Table

Suppose you have a PeopleSoft run control table named PS_AEREQUESTTBL that contains the following columns. You want to have the following run control data. The agent that runs the PeopleSoft job you define is named PSAGENT. The process name you want the job to run is called XYZ and its process type is Application Engine.

Column	Data type	Value
OPRID	VARCHAR2(30)	VP1
RUN_CNTL_ID	VARCHAR2(30)	test
AE_APPLID	VARCHAR2(12)	BI_WF_0001
CURTEMPINSTANCE	NUMBER(38)	0

Column	Data type	Value
PROCESS_FREQUENCY	VARCHAR2(1)	O
AE_PROCESS_STATUS	VARCHAR2(1)	N
PROCESS_INSTANCE	NUMBER(10)	0
PROCESS_ORIG	VARCHAR2(1)	N
LAST_RUN_DTM	DATE	null
MARKET	VARCHAR2(3)	b\
ASOF_DT	DATE	2007-07-10 14:30:00

To add a row of process parameters to a run control table

1. Enter the following required information in the Basic page:
   - Agent name—PSAGENT
   - Process name—XYZ
   - Process type—Application Engine
2. Enter the following required information in the Other PS Parameters page:
   - PS operator ID—VP1
   - Run control ID—test
3. Open the Run Control Table List page.
4. Click Add and enter the following information in the first row of the table:
   - Run control table—PS_AEREQUESTTBL
   - Run control arguments—BI_WF_0001,0,O,N,0,N,,b\",sysdate
5. Click OK.

This request adds the new row of values to the PS_AEREQUESTTBL run control table.

More information:

PeopleSoft Jobs (see page 350)
Distribute PeopleSoft Reports (see page 357)
Using a Namespace for a User that has Different Passwords (see page 63)

Distribute PeopleSoft Reports

If you select EMAIL or WEB as the Output destination type, you can distribute a PeopleSoft report electronically to operators, groups of people, or individuals. Use the Distribution page to specify your email distribution list, an email subject, and message text.

Follow these steps:

1. Open the PeopleSoft job that runs the report you want to distribute.

   The Basic page of the PeopleSoft dialog opens.

2. Click Distribution in the left pane.

   The Distribution page opens in the right pane.

   Note: To enable the fields on this page, you must select EMAIL or WEB as the Output destination type on the Other PS Parameters page.

3. Specify the following distribution information:

   Folder name

   (Optional) Defines a report folder name for the contents of this distribution list.

   Example: Month-end

   Use default distribution

   (Optional) Sends the output report to the operator ID you specified in the PS operator ID field in the Other PS Parameters page.

   Note: Select this option to avoid specifying a distribution list of operator IDs.

   User distribution list

   (Optional) Specifies an operator ID to send the output report to.

   Roles distribution list

   (Optional) Specifies the name to represent the role of the individual who is receiving the report.

   Email with log

   (Optional) Emails job logs along with the report to a recipient on the distribution list.

   Email web report

   (Optional) Emails a web report along with the report to a recipient on the distribution list.

   Email subject

   (Optional) Defines the subject title.

   Example: Month-end report

Message text

(Optional) Defines the email body text.

Example: The report is available for distribution

Email address list

(Optional) Specifies the email address for a recipient on a distribution list.

Example: jsmith@anycompany.com

4. Click OK.

   The PeopleSoft report distribution information is defined.

Remote Execution Jobs

Remote Execution jobs let you run commands on a remote UNIX, HP Integrity NonStop, and OpenVMS computer through Secure Shell (SSH2) or Telnet.

Note: To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Remote Execution. The CA WA Agent for Remote Execution supports only remote target environments of UNIX, HP Integrity NonStop, and OpenVMS.

Define a Remote Execution Job

You can define a Remote Execution (PROXY) job to run commands to a remote UNIX, HP Integrity NonStop, and OpenVMS computer through Telnet or Secure Shell (SSH2).

Notes:

- HP-UX supports only SSH2 protocol.
- To run these jobs, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Remote Execution. The CA WA Agent for Remote Execution supports only remote target environments of UNIX, HP Integrity NonStop, and OpenVMS.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Remote Execution job in the Palette view, and drag the job to the workspace.

   The Remote Execution icon appears on the Application workspace view.

3. Right-click the Remote Execution icon, and select Edit from the pop-up menu.

   The Basic page of the Remote Execution dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that runs the Remote Execution job.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Remote target

   Specifies the name of the custom properties (remote_target.properties) file that is created on the agent for the remote system. The file contains the default user credentials that are used to monitor all jobs run on the target machine by the agent. The name should not include the .properties extension.

   Limits: Up to 128 characters; case-sensitive; valid characters are letters, numbers, and underscore (_); the first character must be a letter

5. Click Add in the Remote commands section, and specify a command or script to run on the remote computer.
6. (Optional) Repeat the previous step to specify additional commands or scripts to run.

   Note: You cannot run multiple UNIX commands in a Remote Execution job.
7. (Optional) Specify the following additional information:

   User

   Specifies the user ID that has the authority to run the job. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Notes:

   - The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
   - Your agent administrator can specify a default setting for all Remote Execution jobs by setting the default.user property in the custom properties file for the remote system. The value you specify in the job definition overrides the default value in the custom properties file.

   Spool file

   Specifies the path to the spool file when you do not want to use the default location. If a spool file is specified, it overwrites a previous spool file with the same name unless you specify to append by beginning the name with >>.

   Limits: Up to 256 characters; case-sensitive; valid characters are a-z, A-Z, 0-9, and delimiters

   Notes:

   - Your agent administrator can specify a default setting for all Remote Execution jobs by setting the spoolHome property in the custom properties file for the remote system. The value you specify in the job definition overrides the default value in the custom properties file.
   - The operating system or file system may impose additional requirements on the value, such as using valid path and file name notation.

8. (Optional) Click Submit Modifier in the left pane to specify commands that you want to run prior to executing the commands on the remote system. Running these setup commands provides additional flexibility to pass additional information to commands executed on the target system.

   The Submit Modifier page opens in the right pane.

9. Click Add, and then enter the appropriate command or option.

   Note: The syntax and option requirements of the target system job apply.

10. (Optional) Repeat the previous step to specify additional setup commands.

11. Click OK.

    The Remote Execution job is defined.

Example: Run the ls Command on a Remote System

This example executes the ls command on the hpserver remote system. Additionally, it specifies a spool file location other than the agent default without overwriting previous spool files with the same name.

To run the command on a Remote System

1. Enter the following required information in the Basic page:
   - Name—ls_cmd
   - Agent name—PROXYAGENT
   - Remote command—ls
   - Remote target—hpserver

     Note: A properties file with the same name as the Remote target must exist on the agent machine.
2. Enter **>>/home/user1/test/ls.out** in the Spool file field.
3. Click Submit modifier in the left pane.
4. Click Add, and then enter **pwd;set>/u1/user1/setTest.out** in the table.
5. Click OK.

More information:

Remote Execution Jobs (see page 358)

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-BW Info Package

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

SAP-BW Process Chain

Creates Process Chains on the SAP system.

SAP-Data Archiving

Stores information in an SAP Archiving Object.

SAP-Event

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-R3

Schedules an SAP R/3 job on your SAP system.

Set Up a Connection to Your SAP System

You can configure the Define perspective to connect directly to your SAP system interface.

Follow these steps:

1. Select Window, Preferences from the main menu.

   The Preferences dialog opens.

2. Click Desktop Client, Define Perspective, SAP GUI Parameters in the left pane.

   The SAP GUI Parameters page opens in the right pane.

3. Complete the fields as appropriate:

   User ID

   Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Limits: 32 or fewer characters

   Examples: Bob, Production:Bob

   Notes:

   - The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
   - This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

   Password

   Specifies your SAP connection password.

   RFC Destination

   Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

   Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

   Client

   Specifies a three-digit number that identifies the SAP client.

   Example: 800

   Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Path to sapshcut.exe

Specifies the executable file name (with the entire path) from the installed SAP interface that CA WA Desktop Client invokes to create or modify a variant.

Default: c:\Program Files\SAPpc\sapgui\sapshcut.exe

4. Click OK.

   The connection to your SAP system is set.

Filtering and Listing Jobs on Your SAP System

You can filter and list jobs running on your SAP system, copy them into your Application, and schedule them on the CA Workload Automation DE server. Reusing jobs saves time and effort. You can reuse existing jobs and modify their properties as required instead of defining the properties of new jobs from scratch.

Your SAP system may contain thousands of individual jobs. You can save time by creating filters for jobs you search for and work with frequently. For example, you might want to create filters for jobs with specific names created by specific users. Using a job filter also lets your system return the requested jobs more quickly and with a lower probability of disturbing system performance.

Note: The CA Workload Automation DE server may need to connect to an FTP server on the agent site depending on the amount of data it retrieves. If your administrator has not configured your user ID and password for the FTP site, or it changed, you are prompted for it.

You can filter and list SAP-R3, SAP-Batch Input Session (BDC), SAP-Business Warehouse Info Package, and SAP-Business Warehouse Process Chain jobs.

Filter and List SAP-Batch Input Session Jobs

You can filter and list SAP-Batch Input Session jobs running on your SAP system, copy them into your Application, update them as required, and schedule them on the server.

Follow these steps:

1. Open the Define perspective.
2. Click the drop-down arrow beside the SAP Views icon in the menu bar, and select BDC Select from the drop-down menu.

   The BDC Select dialog opens.
3. Right-click in the filter listing, and select Add from the pop-up menu.

   The New filter dialog opens.
4. Complete the following fields as appropriate:

   Filter name

   Defines the name of a new filter.

   Connection name

   Indicates the active server connection.
5. Complete the following fields in the SAP Logon information section as appropriate:

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   RFC Destination

   Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

   Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

6. Complete the following required fields in the Filtering criteria section as appropriate:

SAP job name

Specifies the SAP job name that you want to filter on.

Note: You must include a wildcard for a partial name. For example, PROD* matches all job names that start with PROD.

SAP user name

Specifies the SAP user name that you want to filter on.

Note: You must include a wildcard for a partial name. For example, CYB* matches all user names that start with CYB.

7. (Optional) Specify the advanced filtering options with the following steps:
   a. Click Advanced.

      The Advanced filtering criteria dialog opens.

   b. Complete the fields as appropriate.

      Note: Your SAP system may contain thousands of individual jobs. The more specific the filtering criteria, the more fitting the returned jobs are.

   c. Click OK.
8. Click Save and List.

   A list of SAP-Batch Input Session jobs that satisfy your filter is generated.

More information:

Filtering and Listing Jobs on Your SAP System (see page 364)

Filter and List SAP-BW Info Package Jobs

You can filter and list SAP-BW Info Package jobs running on your SAP system, copy them into your Application, update them as required, and schedule them on the server.

Follow these steps:

1. Open the Define perspective.
2. Click the drop-down arrow beside the SAP Views icon in the menu bar, and select BWIP Select from the drop-down menu.

   The BWIP Select dialog opens.
3. Right-click in the filter listing, and select Add from the pop-up menu.

   The New filter dialog opens.
4. Complete the following fields as appropriate:

   Filter name

   Defines the name of a new filter.

Connection name

Indicates the active server connection.

5. Complete the following fields in the SAP Logon information section as appropriate:

Agent name

Specifies the name of the SAP Agent.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

RFC Destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

6. Complete the following fields in the BW Info Package search criteria section as appropriate:

Description from

Defines the range of the description selection criteria.

Info Source from

Defines the range of the info source selection criteria.

Source System from

Defines the range of the source system selection criteria.

Data Source from

Defines the range of the data source selection criteria.

Note: You can use the corresponding field to specify the high value of a range. For example, an info source range of A* to C* returns all Business Warehouse Info Packages that start with A, B, or C.

7. Click Save and List.

   A list of SAP-BW Info Package jobs that satisfy your filter is generated.

More information:

Filtering and Listing Jobs on Your SAP System (see page 364)

Filter and List SAP-BW Process Chain Jobs

You can filter and list SAP-BW Process Chain jobs running on your SAP system, copy them into your Application, update them as required, and schedule them on the server.

Follow these steps:

1. Open the Define perspective.
2. Click the drop-down arrow beside the SAP icon in the menu bar, and select BWPC Select from the drop-down menu.

   The BWPC Select dialog opens.
3. Right-click in the filter listing, and select Add from the pop-up menu.

   The New filter dialog opens.
4. Complete the following fields as appropriate:

   Filter name

   Defines the name of a new filter.

Connection name

Indicates the active server connection.

5. Complete the following fields in the SAP Logon information section as appropriate:

Agent name

Specifies the name of the SAP Agent.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

RFC Destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

6. Complete the following fields in the BW Process Chain search criteria section as appropriate:

Process Chain name

(Optional) Specifies the process chain name that you want to filter on.

Note: You must include a wildcard for a partial name. For example, PC* matches all process chains that start with PC.

Process description

(Optional) Specifies the process description that you want to filter on.

Note: You must include a wildcard for a partial name. For example, PC_* matches all process descriptions that start with PC_.

7. Click Save and List.

   A list of SAP-BW Process Chain jobs that satisfy your filter is generated.

More information:

Filtering and Listing Jobs on Your SAP System (see page 364)

Filter and List SAP-R3 Jobs

You can filter and list SAP-R3 jobs running on your SAP system, copy them into your Application, update them as required, and schedule them on the server.

Follow these steps:

1. Open the Define perspective.
2. Click the drop-down arrow beside the SAP Views icon in the menu bar, and select SAP R3 Select from the drop-down menu.

   The SAP R3 Select dialog opens.
3. Right-click in the filter listing, and select Add from the pop-up menu.

   The New filter dialog opens.
4. Complete the following fields as appropriate:

   Filter name

   Defines the name of a new filter.

   Connection name

   Indicates the active server connection.
5. Complete the following fields in the SAP Logon information section as appropriate:

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

RFC Destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

6. Complete the following fields in the Filtering criteria section as appropriate:

SAP job name

Specifies the SAP job name that you want to filter on.

Note: You must include a wildcard for a partial name. For example, PROD* matches all job names that start with PROD.

SAP user name

Specifies the SAP user name that you want to filter on.

Note: You must include a wildcard for a partial name. For example, CYB* matches all user names that start with CYB.

7. (Optional) Specify advanced filtering options with the following steps:
   a. Click Advanced.

      The Advanced filtering criteria dialog opens.

   b. Complete the fields as appropriate.

      Note: Your SAP system may contain thousands of individual jobs. The more specific the filtering criteria, the more fitting the returned jobs are.

   c. Click OK.
8. Click Save and List.

   A list of SAP-R3 jobs that satisfy your filter is generated.

Example: Use Advanced Filtering

Suppose that you want to list only jobs that finished running between June 1, 2006 and June 30, 2006, between the hours of 12 p.m. and 5 p.m.

To use advanced filtering

1. Open the Advanced filtering criteria dialog.
2. Select the From date check box, and select June 1, 2006 from the From date drop-down list using the calendar.
3. Select the From time check box, and select 12:00:00 PM from the From time text box using the up and down arrows.
4. Select the To date check box, and select June 30, 2006 from the To date drop-down list using the calendar.
5. Select the To time check box, and select 5:00:00 PM from the To time text box using the up and down arrows.
6. Click OK.

More information:

Filtering and Listing Jobs on Your SAP System (see page 364)

Copy Jobs Defined on Your SAP System to Your Application

After defining a filter, you can list the jobs on the SAP system that match your filter criteria. You can then select the jobs you want to reuse, copy them into your Application, and update their job definitions as required.

Follow these steps:

1. Filter and list the relevant jobs in the Define perspective.

   A list of SAP jobs that satisfy your filter is generated.

2. Select the job you want to copy into your Application, and drag and drop the job onto the Application workspace.

   The job definition is retrieved from the SAP system and the appropriate icon is added to the Application workspace.

3. Right-click the SAP icon, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

4. Modify the fields as required, and click OK.

   The SAP job is copied to your Application.

More information:

Filtering and Listing Jobs on Your SAP System (see page 364)

Define an SAP-Batch Input Session Job

You can define an SAP-Batch Input Session 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-Batch Input Session job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-Batch Input Session icon appears on the Application workspace view.

3. Right-click the SAP-Batch Input Session icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-Batch Input Session dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Job name

   Specifies the name of the ABAP job that creates the BDC session on the SAP system.

   Note: You can view this job name on the SAP Job Overview window.

Release Option

Specifies how the job is released, as follows:

- Immediately—Indicates that the job is released immediately. If no free background processing is available, the job is not released and stays in the SAP job state Scheduled.
- As soon as possible—Indicates that the job is released as soon as possible.
- Do not release—Indicates that the job is not released. This option requires a manual process to release the job. When the job is released, the agent detects the changed status and displays it.

Default: As soon as possible

5. (Optional) Specify the following additional information:

RFC destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Target SAP System

Specifies the name of the SAP system where this job runs. The target system identifies the name of the host server where SAP runs.

Note: To display a list of available SAP systems, click Refresh to the right of this field.

Background destination

Specifies the name of the background server that will process the Batch Input Session.

Minimum processed rate

Specifies the minimum acceptable process rate as a percentage of the total transactions. For example, a processed rate of 95 means that the SAP-Batch Input Session job fails if less than 95 percent of transactions are processed.

Maximum error rate

Specifies the maximum acceptable error rate as a percentage of the total transactions. For example, an error rate of 2 means that the SAP-Batch Input Session job fails if more than 2 percent of transactions contain errors.

Extended log

Indicates that the job generates advanced logging of the Batch Input Session.

6. Click Step Specifications in the left pane.

The Step Specifications page opens in the right pane.

7. Click Add and complete the following fields as appropriate to specify an ABAP (step):

Name

Specifies the ABAP name.

Note: To display the ABAPs defined on the SAP system, click View ABAPs. In the Get ABAP List dialog, enter the search criteria in the ABAP name field and click Refresh. You can select the ABAP from the list.

Variant

(Optional) Specifies a variant to pass to the ABAP.

Notes:

- To view the variants defined on the SAP system, click View Variants. In the Get Variant List for ABAP dialog, enter the search criteria in the Name field and click Refresh. You can select the variant from the list. You can also edit the variant by clicking Edit variant.
- When retrieving the variant details, SAP does not return values for the following fields:
  - Only for background processing
  - Protect variant
  - Only display in catalog

Step user

(Optional) Specifies the user ID for the step.

Default: The user specified in the User name field on the Basic page

Language

(Optional) Specifies the language to use to log on to the SAP system.

Example: EN for English

The ABAP is added to the Steps table.

Notes:

- To update an ABAP (step), select the row in the Steps table and modify the fields in the ABAP program as required.
- To delete an existing ABAP (step), select the row in the Steps table and click Delete.
- To specify print or archive parameters for an ABAP, select a row in the Steps table, and click Print Specifications. Specify the parameters in the Background Print Parameters dialog and click OK to save the parameters for the current selected step.
- To send the SAP spool file by email on step completion or failure, select a row in the Steps table, and click Additional Attributes. Specify the email recipients in the Step Attributes dialog and click OK to save the recipients for the current selected step.

8. Click OK.

   The SAP-Batch Input Session job is defined.

Define an SAP-BW Info Package Job

You can define an SAP-BW Info Package 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-BW Info Package job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-BW Info Package job icon appears on the Application workspace view.

3. Right-click the SAP-BW Info Package icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-BW Info Package dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Info package

   Specifies the name of the Info Package.

   Note: To display a list of Info Packages defined on the SAP system, click Refresh. To modify an Info Package, specify an Info Package in this field and click Details. On the Info Package Details dialog, modify the Info Package and click OK. When the job runs, the server modifies the Info Package before submitting the Info Package job on the SAP system.

5. (Optional) Specify the following additional information:

RFC destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Job name

Specifies the name of the Business Warehouse Info Package job on the SAP system.

6. Click OK.

The SAP-BW Info Package job is defined.

Define an SAP-BW Process Chain Job

You can define an SAP-BW Process Chain job to run a sequence of background processes on the SAP system. Some SAP processes trigger events that can start other processes. An SAP-BW 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-BW Process Chain job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-BW Process Chain job icon appears on the Application workspace view.

3. Right-click the SAP-BW Process Chain icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-BW Process Chain dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Chain identifier

   Specifies the name of the Process Chain.

   Note: To display a list of Process Chains defined on the SAP system, click Refresh.

5. (Optional) Specify the following additional information:

   RFC destination

   Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

   Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

6. Click OK.

   The SAP-BW Process Chain job is defined.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)

Define an SAP-Data Archiving Job

You can define an SAP-Data Archiving 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-Data Archiving job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-Data Archiving job icon appears on the Application workspace view.

3. Right-click the SAP-Data Archiving icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-Data Archiving dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Archiving object

   Specifies the name of the Archiving object.

   Note: To display a list of Archiving objects defined on the SAP system, click Refresh to the right of this field.

   Variant

   Specifies the name of the Archiving object variant.

   Note: To display a list of Archiving objects variants defined on the SAP system, click Refresh to the right of this field.

5. (Optional) Specify the following additional information:

RFC destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Target SAP system

Specifies the name of the SAP system where this job runs. The target system identifies the name of the host server where SAP runs.

Note: To display a list of available SAP systems, click Refresh to the right of this field.

6. Click Print Specifications.

The Background Print Parameters dialog opens.

7. Complete the following required fields as appropriate:

   Output device

   Specifies the name of the output device.

   Note: To display a list of available printers, click Refresh to the right of this field.

   Archiving mode

   Indicates whether you want to specify only print parameters, only archive parameters, or both print and archive parameters.

   Default: Print only

   Note: If you select the Archive only or Print and archive options, you must also click Archive Parameters and enter values in the Object type, Document type, and Information fields in the Background Archive Parameters dialog.

8. (Optional) Complete the remaining fields in the dialog as appropriate.
9. (Optional) Click Archive Parameters, complete the remaining fields in the dialog as appropriate, and click OK.
10. Click OK twice.

    The SAP-Data Archiving job is defined.

Define an SAP-Event Job

You can define an SAP-Event 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-Event job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-Event job icon appears on the Application workspace view.

3. Right-click the SAP-Event icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-Event dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Name (SAP Event information section)

   Specifies the name of the SAP event.

   Action

   Indicates whether the job monitors an SAP event or triggers an SAP event.

   Default: Monitor for SAP Event

5. (Optional) Specify the following additional information:

RFC Destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Parameter

Specifies the name of the event parameter such as job name or job count.

Note: CA Workload Automation DE and SAP consider event names with different event parameters to be different events.

Alert

Specifies the name of the Alert that the server triggers when the job detects the triggering (raising) of the SAP event.

Note: This field applies only to the monitoring of SAP Events. If you specify an Alert, the job monitors continuously and the server sends the Alert every time the job detects the triggering (raising) of the SAP event.

6. Click OK.

   The SAP-Event job is defined.

More information:

Continuous Monitoring Usage (see page 64)
Using a Namespace for a User that has Different Passwords (see page 63)

Define an SAP-Job Copy Job

You can define an SAP-Job Copy 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-Job Copy job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-Job Copy job icon appears on the Application workspace view.

3. Right-click the SAP-Job Copy icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-Job Copy dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the SAP Agent.

Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Job count

Specifies the job count of the SAP-R3 job you want to copy.

Job name

Specifies the name of the SAP-R3 job you want to copy.

Release Option

Specifies how the job is released, as follows:

- Immediately—Indicates that the job is released immediately. If no free background processing is available, the job is not released and stays in the SAP job state Scheduled.
- As soon as possible—Indicates that the job is released as soon as possible.
- Do not release—Indicates that the job is not released. This option requires a manual process to release the job. When the job is released, the agent detects the changed status and displays it.

Default: As soon as possible

5. (Optional) Specify the following additional information:

RFC destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Step number

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.

Target job name

Specifies the name of the target (new) job.

Default: Source job specified in the Name field

Target SAP system

Specifies the name of the SAP system where this job runs. The target system identifies the name of the host server where SAP runs.

Note: To display a list of available SAP systems, click Refresh to the right of this field.

Success Message

Defines the text string or regular expression to match within the job's output file to indicate job success. If the success message appears in the output file, the job will be marked as completed even if it fails on the SAP system. The next job is released.

Failure Message

Defines the text string or regular expression to match within the job's output file to indicate job failure. If the failure message appears in the output file, the job will be marked as failed even if it succeeds on the SAP system. The next job is not released.

Web posting

Indicates whether to post the SAP job log and spool on the web.

Monitor children

Indicates whether to monitor the children of the job. The job completes successfully only if it and all of its children complete successfully.

6. Click OK.

The SAP-Job Copy job is defined.

Define an SAP-Process Monitor Job

You can define an SAP-Process Monitor 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-Process Monitor job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-Process Monitor job icon appears on the Application workspace view.

3. Right-click the SAP-Process Monitor icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-Process Monitor dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Process status

   Indicates the status of the processes to be monitored (Waiting, Running, or Stopped).

5. (Optional) Specify the following additional information:

   RFC Destination

   Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

   Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Process user

Specifies an SAP user to associate with the job the process runs when a process matches the specified process status.

Note: This field does not apply to the Waiting process status.

Process client

Specifies the client monitored for the Running or Stopped process status.

Note: This field does not apply to the Waiting process status.

Process type

Indicates the SAP business process type to monitor. Select one of the following process types:

- DIA—dialog
- UPD—update
- ENQ—lock
- BGD—background
- SPO—spool
- UP2—update2

Target SAP system

Specifies the name of the SAP system where this job runs. The target system identifies the name of the host server where SAP runs.

Note: To display a list of available SAP systems, click Refresh to the right of this field.

ABAP

Specifies the name of the ABAP program on the SAP system.

Note: This field does not apply to the Waiting process status.

Alert

Specifies the name of the Alert that the CA Workload Automation DE server triggers when the specified process status and type are detected.

Note: If you specify an Alert, the job monitors continuously and the server sends the Alert every time the job detects the triggering (raising) of the SAP event.

6. Click OK.

   The SAP-Process Monitor job is defined.

More information:

Continuous Monitoring Usage (see page 64)
Using a Namespace for a User that has Different Passwords (see page 63)

Define an SAP-R3 Job

You can define an SAP-R3 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SAP-R3 job from the SAP group in the Palette view, and drag the job to the workspace.

   The SAP-R3 job icon appears on the Application workspace view.

3. Right-click the SAP-R3 icon, and select Edit from the pop-up menu.

   The Basic page of the SAP-R3 dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the SAP Agent.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Job name

   Specifies the SAP job name that identifies the workload within the SAP system.

   Limits: 32 alphanumeric characters; can include spaces, periods, and the characters @, $, and #

Release Option

Specifies how the job is released, as follows:

- Immediately—Indicates that the job is released immediately. If no free background processing is available, the job is not released and stays in the SAP job state Scheduled.
- As soon as possible—Indicates that the job is released as soon as possible.
- Do not release—Indicates that the job is not released. This option requires a manual process to release the job. When the job is released, the agent detects the changed status and displays it.

Default: As soon as possible

5. (Optional) Specify the following additional information:

RFC Destination

Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Target SAP System

Specifies the name of the SAP system where this job runs. The target system identifies the name of the host server where SAP runs.

Note: To display a list of available SAP systems, click Refresh to the right of this field.

Success Message

Defines the text string or regular expression to match within the job's output file to indicate job success. If the success message appears in the output file, the job will be marked as completed even if it fails on the SAP system. The next job is released.

Failure Message

Defines the text string or regular expression to match within the job's output file to indicate job failure. If the failure message appears in the output file, the job will be marked as failed even if it succeeds on the SAP system. The next job is not released.

Web posting

Indicates whether to post the SAP job log and spool on the web.

Monitor children

Indicates whether to monitor the children of the job. The job completes successfully only if it and all of its children complete successfully.

6. Click Step Specifications in the left pane.

The Step Specifications page opens in the right pane.

7. Click Add and complete the following fields as appropriate to specify an ABAP (step).

Name

Specifies the ABAP name.

Note: To display the ABAPs defined on the SAP system, click View ABAPs. In the Get ABAP List dialog, enter the search criteria in the ABAP name field and click Refresh. You can select the ABAP from the list.

Variant

(Optional) Specifies a variant to pass to the ABAP.

Notes:

- To view the variants defined on the SAP system, click View Variants. In the Get Variant List for ABAP dialog, enter the search criteria in the Name field and click Refresh. You can select the variant from the list. You can also edit the variant by clicking Edit variant.
- When retrieving the variant details, SAP does not return values for the following fields:
  - Only for background processing
  - Protect variant
  - Only display in catalog

Step user

(Optional) Specifies the user ID for the step.

Default: The user specified in the User name field on the Basic page

Language

(Optional) Specifies the language to use to log on to the SAP system.

Example: EN for English

The ABAP is added to the Steps table.

Notes:

- To update an ABAP (step), select the row in the Steps table and modify the fields in the ABAP program as required.
- To create a copy of an existing ABAP (step), select a row in the Steps table and click Copy. Modify the fields in the ABAP program as required.
- To delete an existing ABAP (step), select a row in the Steps table and click Delete.
- To specify print or archive parameters for an ABAP, select a row in the Steps table, and click Print Specifications. Specify the parameters in the Background Print Parameters dialog and click OK to save the parameters for the current selected step.
- To send the SAP spool file by email on step completion or failure, select a row in the Steps table, and click Additional Attributes. Specify the email recipients in the Step Attributes dialog and click OK to save the recipients for the current selected step.

8. (Optional) Repeat the previous step to add additional ABAPs (steps).

   Note: When defining an SAP-R3 job, try to 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 whole job.

9. Click OK.

   The SAP-R3 job is defined.

Using Success and Failure Messages within an SAP Job Definition

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 success message or failure message fields within 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.

For more flexibility, you can specify regular expressions instead of simple text strings within the success message and failure message fields. 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, you 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. To check multiple lines in the file, you must use (?s). The first .* indicates that any number of characters can precede TEST. The second .* indicates that any number of characters can follow it.

  ```
  (?s).*TEST.*
  ```

- This expression checks whether "not found" or "started" appears in the job output file.

  ```
  <(?s).*[not found|started].*>
  ```

- This expression checks whether "Job canceled" appears in the job output file.

  ```
  (?s).*Job\scanceled.*
  ```

Define Recipients for the Job's Output

You can define recipients for the job's output. The recipient can be an email address, an SAPoffice distribution list, or an SAPoffice user.

Follow these steps:

1. Open the SAP-R3 job or SAP-Batch Input Session job for which you want to define recipients for the job's output.

   The Basic page of the job definition dialog opens.

2. Click Spool List Recipients.

   The Spool List Recipients Options dialog opens.

3. Complete the following fields as appropriate:

   Save outgoing document to SAPoffice outbox

   (Optional) Indicates whether to deliver the outgoing documents from this job to the outbox of the user specified in the SAP Logon information section on the Basic page.

   Recipient type

   Indicates whether you want to specify an email address, an SAPoffice distribution list, or an SAPoffice user.

   Default: Email

   Express

   (Optional) Indicates whether to send the spool using Express delivery.

   Note: This option does not apply to the email recipient type.

   No Printing

   Indicates whether to prevent the recipient from printing the spool.

   Note: This option does not apply to the email recipient type.

   No Forwarding

   Indicates whether to prevent the recipient from forwarding the spool.

   Note: This option does not apply to the email recipient type.

   Recipient

   Specifies the recipient name and indicates whether to send the job's output using the To, cc, or bcc option.

   Note: To view the SAPoffice distribution lists or SAPoffice users defined on the SAP system, select the appropriate recipient type and click Refresh.

4. Click Add.

   The recipient is added to the List of Recipients table.

5. (Optional) Repeat the previous steps to add additional recipients.
6. Click OK.

   The recipients for the job output are added.

More information:

Define an SAP-Batch Input Session Job (see page 375)

Define Print or Archive Parameters for a Step

You can define print or archive parameters, or both, for a step.

Follow these steps:

1. Open the SAP-R3 job or SAP-Batch Input Session for which you want to define print or archive parameters for a step.
2. Click Step Specifications in the left pane.

   The Step Specifications page opens in the right pane.
3. Select the step in the Steps section.
4. Click Print Specifications.

   The Background Print Parameters dialog opens.
5. Complete the following required field as appropriate:

   Output device

   Specifies the name of the output device.

   Note: To display a list of available printers, click Refresh to the right of this field.
6. Do one of the following:
   - To specify only print parameters, select Print only in the Archiving mode drop-down list, and complete the fields in the Background Print Parameters dialog as appropriate.
   - To specify only archive parameters, select Archive only in the Archiving mode drop-down list, click Archive Parameters, and complete the fields in the Background Archive Parameters dialog. Click OK to close the Background Archive Parameters dialog.
   - To specify print and archive parameters, select Print and archive in the Archiving mode drop-down list, complete the fields in the Background Print Parameters dialog as appropriate, click Archive Parameters, and complete the fields in the Background Archive Parameters dialog as appropriate. Click OK to close the Background Archive Parameters dialog.
7. Click OK.

   The print and archive parameters are defined for the step.

More information:

Define an SAP-Batch Input Session Job (see page 375)

Send the SAP Spool File by Email on Step Completion or Failure

You can send the SAP spool file by email on step completion or failure.

Follow these steps:

1. Open the SAP R3 job or SAP-Batch Input Session job for which you want to send the SAP spool file by email on step completion or failure.

   The Basic page of the job definition dialog opens.

2. Click Step Specifications in the left pane.

   The Step Specifications page opens in the right pane.

3. Select the step in the Steps section.

4. Click Additional Attributes.

   The Step Attributes dialog opens.

5. Specify the email address in the To field, and click Add.

   The email address is added to the To area.

6. (Optional) Repeat the previous step to add additional email recipients.

7. (Optional) Complete the following fields as appropriate:

   Success message

   Defines the text string or regular expression to match within the job's spool file to indicate ABAP success. If the success message appears in the spool file, the ABAP will be marked as completed even if it fails on the SAP system.

   Failure message

   Defines the text string or regular expression to match within the job's spool file to indicate ABAP failure. If the failure message appears in the spool file, the ABAP will be marked as failed even if it succeeds on the SAP system.

   Note: This step does not apply to SAP-Batch Input Session jobs.

8. Click OK twice.

   The SAP spool file is sent by email on step completion or failure.

More information:

Define an SAP-Batch Input Session Job (see page 375)
Using Success and Failure Messages within an SAP Job Definition (see page 401)

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.

Define an SNMP Subscribe Job

You can define an SNMP Subscribe 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SNMP Subscribe job from the SNMP group in the Palette view, and drag the job to the workspace.

   The SNMP Subscribe job icon appears on the Application workspace view.

3. Right-click the SNMP Subscribe icon, and select Edit from the pop-up menu.

   The Basic page of the SNMP Subscribe dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent configured as an SNMP manager.

   MIB

   Specifies the relative or absolute path to and name of the Management Information Base (MIB) file. You can type the path and name of the MIB file or browse for the MIB file on the agent computer using the arrow next to the field. If the MIB file is not loaded successfully, the job fails.

   Filter

   Specifies a regular expression to use to filter SNMP trap information.

   Note: 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.

5. (Optional) Specify the following additional information:

   Output destination

   Specifies the location of the Java serialized object produced by the job.

   Limits: Up to 256 characters

   Note: If you do not specify an output destination, 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.

   Continuous monitoring, using alert

   Specifies the identifier of an Alert to be triggered when an SNMP trap matches the filter condition. Defines the job as continuous. To end continuous monitoring, you must complete the job manually. If you do not specify a value in this field, the job completes when the agent detects the filter value.

6. Click OK.

   The SNMP Subscribe job is defined.

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.

To subscribe to the first SNMP trap the agent receives

1. Enter the following information in the Basic page:
   - Name—SUBSCRIBE_FIRST
   - Agent name—SNMPAGENT
   - MIB—C:\SNMP\MIBs\RFC1213-MIB.mib
   - Filter—.*
2. Click OK.

Note: To continuously subscribe to all SNMP traps, specify an Alert in the Continuous monitoring, using alert field.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
Continuous Monitoring Usage (see page 64)
SNMP Jobs (see page 406)

Define an SNMP Trap Send Job

You can define an SNMP Trap Send 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SNMP Trap Send job from the SNMP group in the Palette view, and drag the job to the workspace.

   The SNMP Trap Send job icon appears on the Application workspace view.

3. Right-click the SNMP Trap Send icon, and select Edit from the pop-up menu.

   The Basic page of the SNMP Trap Send dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent configured as an SNMP manager.

   MIB

   Specifies the relative or absolute path to and name of the Management Information Base (MIB) file. You can type the path and name of the MIB file or browse for the MIB file on the agent computer using the arrow next to the field. If the MIB file is not loaded successfully, the job fails.

   SNMP Object ID

   Specifies the SNMP object identifier (OID) in string format. The OID identifies the trap that you want to send. You can type the OID or browse for an OID in a MIB file using the arrow next to the field.

   Example: cybtrapstart

   Note: You cannot specify the OID in numeric format for SNMP Trap Send jobs.

   Host name

   Specifies the host name or IP address of the network device you want to send the SNMP trap to.

Version

Specifies the SNMP version that is used to generate the SNMP trap. Options are 1, 2, and 3.

Default: 3

5. Specify the following fields if your SNMP version is v3:

SNMP User

Specifies the user name whose credentials are used for authentication. The user must be defined in the Topology.

Authentication protocol

Specifies the SNMP v3 authentication protocol to use when connecting with the user specified in the SNMP User field. Options are the following:

MD5

Specifies the Message Digest 5 (MD5) authentication protocol.

SHA

Specifies the Secure Hash (SHA) authentication protocol.

Privacy protocol

Specifies the SNMP v3 privacy protocol to use. Options are the following:

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).

6. Specify the following attribute if your SNMP version is v1 or v2:

Community

Specifies the community string that is used to authenticate against the network device.

7. (Optional) Specify the following additional information:

Port

Specifies the port of the network device that sends the SNMP trap.

Default: 162

Inform

Indicates that the job sends the SNMP v2/v3 INFORM message if it exists. This option is recommended for SNMP v3.

Engine ID

Specifies the SNMP engine ID, which uniquely identifies an SNMP engine and its corresponding SNMP entity within an administrative domain.

Default: AGENT_ENGINE

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

8. (Optional) Click Parameter List in the left pane to define parameters. The parameters represent the trap variables or objects.

   The Parameter List page opens in the right pane.

9. Click New to define the first parameter.

   Note: If you specify a parameter value, you must specify values for all trap parameters. For example, if the trap contains five parameters, you must specify values for all five parameters. The order of the parameters in the table must match the order of the variables defined in the MIB file for the trap.

   The Add Parameter dialog opens.

10. Select one of the following parameter options:

    - Value—Identifies a single value.
    - Payload producing job—Identifies the name of a predecessor job.

    The fields related to the option appear.

11. Complete the following fields, as required:

    Type

    Specifies the SNMP parameter type. This field is only mandatory for the Value parameter option. Options for the type include the following:

    int

    Specifies an integer value.

    uint

    Specifies an unsigned integer value.

ticks

Specifies a time ticks value (measured in hundreds of a second).

addr

Specifies an IP address.

Note: SNMP only supports IPv4 addresses.

oid

Specifies an object identifier (OID) 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 2^{32} - 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 2^{32} - 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.

Example: 2343567

Value

Specifies the value of the parameter to pass to the trap. This field is used with the Value parameter option.

Note: The value must be in the correct format as specified by the Type field.

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

12. Click OK.

    The Add parameter dialog closes and the parameter appears in the Parameters table.

13. Repeat Steps 9 through 12 to specify values for the remaining parameters.

14. Click OK.

    The SNMP Trap Send job is defined.

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. The cybtrapstart trap contains five variables, so five parameters are passed to the trap. In this example, 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.

To send an SNMP trap to a network device

1. Enter the following information in the Basic page:
   - Name—SENDTRAP
   - Agent name—SNMPAGENT
   - MIB—C:\SNMP\MIBs\RFC1213-MIB.mib
   - SNMP Object ID—cybtrapstart
   - Host name—localhost
   - Port—162
   - Version—3
   - SNMP User—user1
2. Select SHA from the Authentication protocol drop-down list.
3. Select AES from the Privacy protocol drop-down list.

4. Add the following five value parameters in the Parameter List page:
   - Parameter Type—string
   - Parameter Value—p1
   - Parameter Type—string
   - Parameter Value—p2
   - Parameter Type—string
   - Parameter Value—p3
   - Parameter Type—string
   - Parameter Value—p4
   - Parameter Type—string
   - Parameter Value—p5
5. Click OK.

   When the job runs, the SNMP trap is sent.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
SNMP Jobs (see page 406)
Browse the Contents of a MIB File for an SNMP Trap Send Job (see page 415)

Browse the Contents of a MIB File for an SNMP Trap Send Job

You can use the Browse MIB File wizard to browse the contents of a MIB file and populate the MIB and SNMP Object ID fields with valid SNMP data. You can also use the wizard to populate the trap parameters in the Parameter List page.

Follow these steps:

1. Open the Basic page of the SNMP Trap Send job definition.
2. Click the arrow next to the SNMP Object ID field.

   The Load MIB File page of the Browse MIB File wizard opens.
3. Select the MIB file using one of the following options:

   Specify Agent Location

   Specifies the relative or absolute path to and name of a MIB file on the agent computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: This field is populated with the MIB file name specified in the Basic page of the job definition, if any. If you want the changes you make to be reflected in the job definition, select the Set the selected location in the job definition panel check box.

   Specify Location

   Specifies the relative or absolute path to and name of a MIB file on the local computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: SNMP jobs require that the MIB file is stored on the agent computer. If you are working offline, you can browse a local MIB file, but verify that the local MIB file is identical to the agent MIB file. You cannot specify a local MIB file in the job definition.
4. Click Next.

   The specified MIB file is loaded and parsed. If the file is successfully read, the Select SNMP Node page opens.

Note: If you get a warning message about incomplete MIB data, consider the following:

- A MIB file can contain information imported from other MIB files. The IMPORTS section of the MIB file lists the imported modules.
- For local MIB files, CA WA Desktop Client only tries to load the imported MIB files that are in the *same* directory as the current MIB file. If the files are in a different location, the import dependencies will not be resolved.
- For agent MIB files, only the current MIB file is retrieved. CA WA Desktop Client does not try to load the imported MIB files because loading all the files could be time consuming.
- As a workaround for unresolved imports, you can copy all the unresolved MIB files on a local computer and browse them using the Specify Location option.

5. Select an SNMP v1 or v2 trap from the MIB tree structure in the left pane. SNMP v1 traps are displayed under a separate root node named SNMP V1 Traps.

   Notes:

   - To expand all nodes in the tree, right-click any node and select Expand All from the pop-up menu.
   - To search for a node in a subtree, right-click the root node of the subtree and select Find in MIB subtree from the pop-up menu. The search is based on the full name of the node. You cannot use wildcards or specify a partial name. If a matching node in the subtree is found, the node is selected in the tree. You cannot search for nodes under the SNMP V1 Traps tree.

   The properties of the selected node appear in the right pane.

6. (Optional) Click Next to view the parameters for the selected node. The parameters represent the trap variables or objects.

   The Define SNMP parameters page opens.

7. (Optional) Select a parameter from the SNMP parameters table and click Edit to specify the type and value for the selected parameter.

   Note: If you specify a parameter value, you must specify values for all trap parameters. For example, if the trap contains five parameters, you must specify values for all five parameters. The order of the parameters in the table must match the order of the variables defined in the MIB file for the trap.

   The Edit parameter dialog opens.

8. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.

   The fields related to the option appear.

9. Complete the following fields, as required:

Type

Specifies the SNMP parameter type. This field is only mandatory for the Value parameter option. Options for the type include the following:

int

Specifies an integer value.

uint

Specifies an unsigned integer value.

ticks

Specifies a time ticks value (measured in hundreds of a second).

addr

Specifies an IP address.

Note: SNMP only supports IPv4 addresses.

oid

Specifies an object identifier (OID) 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 2^{32} - 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 2^{32} - 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.

Example: 2343567

Value

Specifies the value of the parameter to pass to the trap. This field is used with the Value parameter option.

Note: The value must be in the correct format as specified by the Type field.

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

10. Click OK.

    The Edit parameter dialog closes and the updated parameter appears in the SNMP parameters table.

11. Repeat Steps 7 through 10 to specify values for the remaining parameters.

12. Click Finish.

    The Browse MIB File wizard closes. The SNMP Trap Send job definition is populated with data specified using the wizard.

Define an SNMP Value Get Job

You can define an SNMP Value Get 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SNMP Value Get job from the SNMP group in the Palette view, and drag the job to the workspace.

   The SNMP Value Get job icon appears on the Application workspace view.

3. Right-click the SNMP Value Get icon, and select Edit from the pop-up menu.

   The Basic page of the SNMP Value Get dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

Agent name

Specifies the name of the agent configured as an SNMP manager.

SNMP Object ID

Specifies the SNMP object identifier (OID) in numeric or string format. The OID identifies the variable whose value you want to retrieve. You can type the OID or browse for an OID in a MIB file using the arrow next to the field.

Examples: agentLogLevel, .1.3.6.1.4.1.11203.7.5.0

Host name

Specifies the host name or IP address of the network device that hosts the MIB variable.

Version

Specifies the SNMP version that is used to connect to the network device.

Limits: 1, 2, or 3

Default: 3

5. Specify the following fields if your SNMP version is v3:

SNMP User

Specifies the user name whose credentials are used for authentication. The user must be defined in the Topology.

Authentication protocol

Specifies the SNMP v3 authentication protocol to use when connecting with the user specified in the SNMP User field. Options are the following:

MD5

Specifies the Message Digest 5 (MD5) authentication protocol.

SHA

Specifies the Secure Hash (SHA) authentication protocol.

Privacy protocol

Specifies the SNMP v3 privacy protocol to use. Options are the following:

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).

6. Specify the following attribute if your SNMP version is v1 or v2:

   Community

   Specifies the community string that is used to authenticate against the network device.

7. (Optional) Specify the following additional information:

   MIB

   Specifies the relative or absolute path to and name of the Management Information Base (MIB) file. You can type the path and name of the MIB file or browse for the MIB file on the agent computer using the arrow next to the field. If the MIB file is not loaded successfully, the job fails.

   Port

   Specifies the port of the network device the agent uses for communication.

   Default: 161

   Subtree

   Indicates that the job walks the whole SNMP subtree starting with the OID specified in the SNMP Object ID field.

   Table view

   Indicates that the job retrieves the values of the SNMP subtree in table format. The table is stored as job output variables, which you can use as input to another job in the Application.

   Context name

   Specifies the name of the context that the variable belongs to. 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 Context name field, together with the Context Engine ID field, to uniquely identify a variable within an administrative domain.

   Context Engine ID

   Specifies the context engine ID in hexadecimal format. The context engine ID 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 Context Engine ID field, together with the Context name field, to uniquely identify a variable within an administrative domain.

   Example: X'cab0'

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

8. Click OK.

   The SNMP Value Get job is defined.

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.

To query a network device for the value of an SNMP variable

1. Enter the following information in the Basic page:
   - Name—GETVAR
   - Agent name—SNMPAGENT
   - MIB—C:\SNMP\MIBs\RFC1213-MIB.mib
   - SNMP Object ID—agentVersion
   - Host name—host.example.com
   - Port—161
   - Version—2
   - Community—public
2. Click OK.

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.

To query a network device for the values of a whole SNMP subtree

1. Enter the following information in the Basic page:
   - Name—GETSUBTREE
   - Agent name—SNMPAGENT
   - MIB—C:\SNMP\MIBs\RFC1213-MIB.mib
   - SNMP Object ID—pluginManagerPluginsTable
   - Host name—host.example.com
   - Port—161
   - Version—2
   - Community—public
2. Select the Subtree check box.
3. Click OK.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
SNMP Jobs (see page 406)
Browse the Contents of a MIB File for an SNMP Value Get Job (see page 423)

Browse the Contents of a MIB File for an SNMP Value Get Job

You can use the Browse MIB File wizard to browse the contents of a MIB file and populate the MIB and SNMP Object ID fields with valid SNMP data.

Follow these steps:

1. Open the Basic page of the SNMP Value Get job definition.
2. Click the arrow next to the SNMP Object ID field.

   The Load MIB File page of the Browse MIB File wizard opens.
3. Select the MIB file using one of the following options:

   Specify Agent Location

   Specifies the relative or absolute path to and name of a MIB file on the agent computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: This field is populated with the MIB file name specified in the Basic page of the job definition, if any. If you want the changes you make to be reflected in the job definition, select the Set the selected location in the job definition panel check box.

   Specify Location

   Specifies the relative or absolute path to and name of a MIB file on the local computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: SNMP jobs require that the MIB file is stored on the agent computer. If you are working offline, you can browse a local MIB file, but verify that the local MIB file is identical to the agent MIB file. You cannot specify a local MIB file in the job definition.
4. Click Next.

   The specified MIB file is loaded and parsed. If the file is successfully read, the Select SNMP Node page opens.

Note: If you get a warning message about incomplete MIB data, consider the following:

- A MIB file can contain information imported from other MIB files. The IMPORTS section of the MIB file lists the imported modules.
- For local MIB files, CA WA Desktop Client only tries to load the imported MIB files that are in the *same* directory as the current MIB file. If the files are in a different location, the import dependencies will not be resolved.
- For agent MIB files, only the current MIB file is retrieved. CA WA Desktop Client does not try to load the imported MIB files because loading all the files could be time consuming.
- As a workaround for unresolved imports, you can copy all the unresolved MIB files on a local computer and browse them using the Specify Location option.

5. Select a node with a valid OID from the MIB tree structure in the left pane.

   Notes:

   - To expand all nodes in the tree, right-click any node and select Expand All from the pop-up menu.
   - To search for a node in a subtree, right-click the root node of the subtree and select Find in MIB subtree from the pop-up menu. The search is based on the full name of the node. You cannot use wildcards or specify a partial name. If a matching node in the subtree is found, the node is selected in the tree.

   The properties of the selected node appear in the right pane.

6. (Optional) Select the Use SNMP node OID check box to export the numeric OID to the SNMP Object ID field in the job definition.

   By default, the *name* of the node is exported to the job definition.

7. Click Finish.

   The Browse MIB File wizard closes. The SNMP Value Get job definition is populated with data specified using the wizard.

Define an SNMP Value Set Job

You can define an SNMP Value Set 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the SNMP Value Set job from the SNMP group in the Palette view, and drag the job to the workspace.

   The SNMP Value Set job icon appears on the Application workspace view.

3. Right-click the SNMP Value Set icon, and select Edit from the pop-up menu.

   The Basic page of the SNMP Value Set dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent configured as an SNMP manager.

   SNMP Object ID

   Specifies the SNMP object identifier (OID) in numeric or string format. The OID identifies the variable whose value you want to change. You can type the OID or browse for an OID in a MIB file using the arrow next to the field.

   Examples: agentLogLevel, .1.3.6.1.4.1.11203.7.5.0

   Note: If you specify the OID in string (non-numeric) format, you must specify a value in the MIB field.

   Host name

   Specifies the host name or IP address of the network device that hosts the MIB variable.

   Version

   Specifies the SNMP version that is used to connect to the network device.

   Limits: 1, 2, or 3

   Default: 3

5. Specify the following fields if your SNMP version is v3:

   SNMP User

   Specifies the user name whose credentials are used for authentication. The user must be defined in the Topology.

   Authentication protocol

   Specifies the SNMP v3 authentication protocol to use when connecting with the user specified in the SNMP User field. Options are the following:

   MD5

   Specifies the Message Digest 5 (MD5) authentication protocol.

   SHA

   Specifies the Secure Hash (SHA) authentication protocol.

   Privacy protocol

   Specifies the SNMP v3 privacy protocol to use. Options are the following:

   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).

6. Specify the following field if your SNMP version is v1 or v2:

   Community

   Specifies the community string that is used to authenticate against the network device.

7. (Optional) Specify the following additional information:

   MIB

   Specifies the relative or absolute path to and name of the Management Information Base (MIB) file. You can type the path and name of the MIB file or browse for the MIB file on the agent computer using the arrow next to the field. If the MIB file is not loaded successfully, the job fails.

   Note: This field is required if the OID specified in the SNMP Object ID field is in string (non-numeric) format.

   Port

   Specifies the port of the network device the agent uses for communication.

   Default: 161

Context name

Specifies the name of the context that the variable belongs to. 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 Context name field, together with the Context Engine ID field, to uniquely identify a variable within an administrative domain.

Context Engine ID

Specifies the context engine ID in hexadecimal format. The context engine ID 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 Context Engine ID field, together with the Context name field, to uniquely identify a variable within an administrative domain.

Example: X'cab0'

Output destination

Specifies the location of the Java serialized object produced by the job.

Limits: Up to 256 characters

Note: If you do not specify an output destination, 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.

8. Click Parameter List in the left pane.

   The Parameter List page opens in the right pane.

9. Click New to define the parameter. The parameter represents the variable that gets set when the job runs.

   The Add Parameter dialog opens.

10. Select one of the following parameter options:

    - Value—Identifies a single value.
    - Payload producing job—Identifies the name of a predecessor job.

    The fields related to the option appear.

11. Complete the following fields, as required:

    Type

    Specifies the SNMP parameter type. This field is only mandatory for the Value parameter option. Options for the type include the following:

    int

    Specifies an integer value.

    uint

    Specifies an unsigned integer value.

ticks

Specifies a time ticks value (measured in hundreds of a second).

addr

Specifies an IP address.

Note: SNMP only supports IPv4 addresses.

oid

Specifies an object identifier (OID) 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 2^{32} - 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 2^{32} - 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.

Example: 2343567

Value

Specifies the new value of the variable that you are changing. The agent converts the value to the corresponding type. This field is used with the Value parameter option.

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

12. Click OK.

The SNMP Value Set job is defined.

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.

To change the value of an SNMP variable using SNMP v2

1. Enter the following information in the Basic page:
   - Name—SETV2
   - Agent name—SNMPAGENT
   - MIB—C:\SNMP\MIBs\RFC1213-MIB.mib
   - SNMP Object ID—agentLogLevel
   - Host name—host.example.com
   - Port—161
   - Version—2
   - Community—public
2. Add the following value parameter in the Parameter List page:
   - Parameter Type—int
   - Parameter Value—8
3. Click OK.

   When the job runs, the agentLogLevel variable is changed.

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 the user named user1 are used for authorization.

To change the value of an SNMP variable using SNMP v3

1. Enter the following information in the Basic page:
   - Name—SETV3
   - Agent name—SNMPAGENT
   - MIB—c:\SNMP\MIBs\mymib.mib
   - SNMP Object ID—OUT456789
   - Host name—example.com
   - Version—3
   - SNMP User—user1
   - Context name—contxtname
   - Context Engine ID—X'cab0'
2. Select MD5 from the Authentication protocol drop-down list.
3. Select AES from the Privacy protocol drop-down list.
4. Add the following value parameter in the Parameter List page:
   - Parameter Type—int
   - Parameter Value—8
5. Click OK.

   When the job runs, the OUT456789 variable is changed.

More information:

Payload Producing and Payload Consuming Jobs (see page 71)
SNMP Jobs (see page 406)
Browse the Contents of a MIB File for an SNMP Value Set Job (see page 431)

Browse the Contents of a MIB File for an SNMP Value Set Job

You can use the Browse MIB File wizard to browse the contents of a MIB file and populate the MIB and SNMP Object ID fields with valid SNMP data. You can also use the wizard to populate the variable parameter in the Parameter List page.

Follow these steps:

1. Open the Basic page of the SNMP Value Set job definition.
2. Click the arrow next to the SNMP Object ID field.

   The Load MIB File page of the Browse MIB File wizard opens.
3. Select the MIB file using one of the following options:

   Specify Agent Location

   Specifies the relative or absolute path to and name of a MIB file on the agent computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: This field is populated with the MIB file name specified in the Basic page of the job definition, if any. If you want the changes you make to be reflected in the job definition, select the Set the selected location in the job definition panel check box.

   Specify Location

   Specifies the relative or absolute path to and name of a MIB file on the local computer. You can type the path and name of the MIB file or click Browse to browse for the MIB file.

   Note: SNMP jobs require that the MIB file is stored on the agent computer. If you are working offline, you can browse a local MIB file, but verify that the local MIB file is identical to the agent MIB file. You cannot specify a local MIB file in the job definition.
4. Click Next.

   The specified MIB file is loaded and parsed. If the file is successfully read, the Select SNMP Node page opens.

Note: If you get a warning message about incomplete MIB data, consider the following:

- A MIB file can contain information imported from other MIB files. The IMPORTS section of the MIB file lists the imported modules.
- For local MIB files, CA WA Desktop Client only tries to load the imported MIB files that are in the *same* directory as the current MIB file. If the files are in a different location, the import dependencies will not be resolved.
- For agent MIB files, only the current MIB file is retrieved. CA WA Desktop Client does not try to load the imported MIB files because loading all the files could be time consuming.
- As a workaround for unresolved imports, you can copy all the unresolved MIB files on a local computer and browse them using the Specify Location option.

5. Select a scalar node that has read-write access from the MIB tree structure in the left pane.

   Notes:

   - To expand all nodes in the tree, right-click any node and select Expand All from the pop-up menu.
   - To search for a node in a subtree, right-click the root node of the subtree and select Find in MIB subtree from the pop-up menu. The search is based on the full name of the node. You cannot use wildcards or specify a partial name. If a matching node in the subtree is found, the node is selected in the tree.

   The properties of the selected node appear in the right pane.

6. (Optional) Select the Use SNMP node OID check box to export the numeric OID to the SNMP Object ID field in the job definition.

   By default, the *name* of the node is exported to the job definition.

7. (Optional) Click Next to view the parameter for the selected node. The parameter represents the variable that gets set for the selected node when the job runs.

   The Define SNMP parameters page opens.

8. (Optional) Select the parameter from the SNMP parameters table and click Edit to specify the type and value for the selected parameter.

   The Edit parameter dialog opens.

9. Select one of the following parameter options:

   - Value—Identifies a single value.
   - Payload producing job—Identifies the name of a predecessor job.

   The fields related to the option appear.

10. Complete the following fields, as required:

Type

Specifies the SNMP parameter type. This field is only mandatory for the Value parameter option. Options for the type include the following:

int

Specifies an integer value.

uint

Specifies an unsigned integer value.

ticks

Specifies a time ticks value (measured in hundreds of a second).

addr

Specifies an IP address.

Note: SNMP only supports IPv4 addresses.

oid

Specifies an object identifier (OID) 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 2^{32} - 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 2^{32} - 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.

Example: 2343567

Value

Specifies the new value of the variable that you are changing. The agent converts the value to the corresponding type. This field is used with the Value parameter option.

Payload producing job

Specifies the name of the predecessor job that produced the Java serialized object to be used as an input parameter. This field is used with the Payload producing job parameter option.

Note: The job must be defined within the same Application as the job you are defining.

Example: ANYJOB.URI

11. Click OK.

    The Edit parameter dialog closes and the updated parameter appears in the SNMP parameters table.

12. Click Finish.

    The Browse MIB File wizard closes. The SNMP Value Set job definition is populated with data specified using the wizard.

System Jobs

System jobs let you run programs, commands, or scripts on your operating system or wake up a computer remotely.

You can define the following System jobs:

i5/OS

Lets you run a program or an i5/OS command.

OpenVMS

Lets you run an OpenVMS command file.

Tandem NSK

Lets you run a Tandem NSK command file.

UNIX

Lets you run UNIX scripts or execute commands.

Wake on LAN

Lets you wake up a computer remotely.

Windows

Lets you run Windows command files.

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*.

Define an i5/OS Job

You can define an i5/OS job to schedule workload to run on an i5/OS system. The job can run a program or an i5/OS command. You can run i5/OS jobs in the root file system, open systems file system (QOpenSys), and library file system (QSYS).

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the i5/OS job from the System group in the Palette view, and drag the job to the workspace.

   The i5/OS icon appears on the Application workspace view.

3. Right-click the i5/OS icon, and select Edit from the pop-up menu.

   The Basic page of the i5/OS dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that runs programs or commands on an i5/OS system.

   Action to take

   Indicates whether to run a program or issue a command. Select one of the following options:

   Run program

   Runs a program on an i5/OS system. Enter the program name in the Name (Action to take) field. This is the default.

   Run program in file

   Runs a program from a database file member on an i5/OS system. This is roughly equivalent to running a shell script on a UNIX system or a batch file on a Windows system. Enter the file name and member name containing the program source in the Name (Action to take) field.

Issue command

- Issues a command on an i5/OS system when the i5/OS job runs. Enter the i5/OS command in the Name (Action to take) field.

Name (Action to take)

Specifies the name of the program or i5/OS command that is available for execution on your i5/OS system. You can browse for the program or command file or specify one of the following formats depending on the Action to take selection:

- If you select Run program, use one of the following formats:

  program

  library/*program*

  /QSYS.LIB/*library*.LIB/*program*.PGM

- If you select Run program in file, use one of the following formats:

  file

  file(*member*)

  library/*file*(*member*)

  /QSYS.LIB/*library*.LIB/*file*.FILE/*member*.MBR

- If you select Issue command, use one of the following formats:

  command

  library/*command*

  /QSYS.LIB/*library*.LIB/*command*.CMD

Notes:

- If the library name is specified in the Name (Action to take) field, do not specify a value in the Library name field. If you specify the library name in both fields, you will get a submission error.
- If you do not specify the library name in the Name (Action to take) field or the Library name field, the agent uses the job's library list.

5. (Optional) Specify the following additional information:

Positional parameters

Defines the parameter values you want to pass to the program at the time the program is invoked.

Note: You must specify each parameter in the order it is expected in the program, separating each parameter with a blank space. To pass a parameter containing spaces, enclose it in single quotation marks.

Example: *LIBL 'user 1'

i5/OS user ID

Specifies the i5/OS user ID who has the authority to run the program or command. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: If you do not specify a user ID and a default user ID is defined on the agent, the agent runs the job under the default user ID. Contact your agent administrator about the parameters set in the agentparm.txt file.

6. (Optional) Click Environment in the left pane.

   The Environment page opens in the right pane.

7. Specify the following additional information:

Library name

Specifies the name of the library that contains the program, the source for the program, or the command that you want to run. You can browse for the library name or specify one of the following formats:

- *library*
- /QSYS.LIB/*library*.LIB

Notes:

- Do not specify a value for this field if the library name is already specified in the Name (Action to take) field. If you specify the library name in both fields, you will get a submission error.
- If you do not specify the library name in the Name (Action to take) or Library name fields, the agent uses the job's library list.

Library list

Specifies the names of the libraries that you want to add to the library list. Separate each library name with a space. You can specify up to 25 libraries. The libraries are searched in the same order as they are listed.

Limits: Up to 10 characters per library

Notes:

- This field 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 value in this field overrides the library list specified in the job description for the job.

Current library

Specifies the current library for the job. If you do not specify a current library, the agent uses the current library of the user profile the agent runs under. You can browse for the library name or specify one of the following formats:

- *current_library*
- QSYS.LIB/*current_library*.LIB

Job name

Specifies the name of the i5/OS job. This is the name displayed in the job queue.

Limits: Up to 10 characters; the first character must be alphabetical

Note: If you do not specify a value in this field, the job name defaults to the name specified in the Name field.

Job description

Specifies the name of the job description for the program submitted. The value must be a valid i5/OS job description name. You can browse for the library name or specify one of the following formats:

- *description*
- *library*/*description*
- QSYS.LIB/*library*.LIB/*description*.JOBD

Limits: Up to 60 characters

Note: If you do not specify the library name, the i5/OS system will search for it using the library list for the job.

Job queue

Specfies the name of the job queue for the submitted program. The value must be a valid i5/OS job queue name. You can browse for the job queue name or specify one of the following formats:

- *jobqueue*
- *library/jobqueue*

Limits: Up to 60 characters

Note: If you do not specify the library name, the i5/OS system will search for it using the library list for the job.

i5/OS exit program

Specifies the type of exit code returned by an i5/OS job. Select one of the following options:

***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.

***PROGRAM**

Sends the return code of an OPM program. For example, if your job runs an OPM RPG or OPM COBOL program that contains an exit or return statement, that return code is sent as the exit code.

Note: If the program does not set a return code, the i5 system returns a 0 (zero), which is sent as the exit code. By default, an exit code of zero (0) is interpreted as job success.

Notes:

- Ensure that you specify the appropriate exit code type for the program you are running. If you specify the wrong exit code type (for example, you specify *PROGRAM for an ILE program), the agent sends 0 (zero) as the exit code. By default, 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.

Local data area

Specifies the data for the local data area. 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. The format is the following:

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’

8. Click OK.

The i5/OS job is defined.

Example: Define an i5/OS Job to Run a Program

Suppose that you want to run the program named PARAM under the ASYSOPR user ID. This program takes three parameters in the following order: ABC, C1, and P.

To define an i5/OS job to run a program

1. Enter the following information in the Basic page:
   - Name—PARAM
   - Agent name—AGENT
2. Select the Run program option button and enter the following information in the remaining fields:
   - Name—PARAM
   - Positional parameters—ABC C1 P
   - I5/OS User ID—ASYSOPR
3. Click OK.

Example: Define an i5/OS Job to Issue a Command

Suppose that you want to issue the DISPLIB command to display the job's library list. When the job completes, you can view the library list in the job's spool file.

To define an i5/OS job to issue a command

1. Enter the following information in the Basic page:
   - Name—DISPLIB
   - Agent name—AGENT
2. Select the Issue command option button and enter **DSPLIBL** in the Name field.
3. Enter the following information in the Environment page:
   - Library name—*LIBL
   - Current library—ESPAGENT
   - Job name—PAYJOBA
   - Job description—PRDLIB.PRDOPR
   - Job queue—ABCSYS.PAYROLL
4. Click OK.

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.

To specify data for the local data area in hexadecimal format

1. Enter the following information in the Basic page:
   - Name—I5JOB_LDA
   - Agent name—I5AGENT
2. Select the Run program option button and enter the following information in the remaining fields:
   - Name—IVP
   - Local data area—X'C1C2C3C4'
3. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)
i5/OS Jobs (see page 435)

Use a User's Library List

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

Follow these steps:

1. Open the i5/OS job for which you want to use a user's library list.

   The Basic page of the i5/OS dialog opens.
2. Click Environment in the left pane.

   The Environment page opens in the right pane.
3. Complete one of the following options:
   - Enter ***USRPRF** in the Current library field so that the job uses the user's current library.
   - Enter values in the following fields so that the job can access the user's library list by using the job description assigned to the user:
     - Library list—*JOBD
     - Job description—*USRPRF
4. Click OK.

Pass Keyword Parameters to the i5/OS SBMJOB Command

When the server submits an i5/OS job to the i5/OS system, the job must pass through the SBMJOB command to execute. Keyword parameters are additional parameters that the server passes to the i5/OS SBMJOB command. You can specify any valid SBMJOB command keyword and value combination. You can also specify multiple combinations.

Follow these steps:

1. Open the i5/OS job for which you want to pass keyword parameters.
2. Click Keyword Parameters in the left pane.

   The Keyword Parameters page opens in the right pane.
3. Enter the following information:

   Keyword

   Specifies a valid SBMJOB command keyword.

   Examples: PRTDEV, OUTQ, or USER

   Value

   Specifies a valid SBMJOB value that corresponds to the SBMJOB command keyword, the value that would appear in brackets after the keyword in the SBMJOB command.
4. Click OK.

   When the job runs, the server adds the keyword and value combinations you specified to the SBMJOB command.

SBMJOB Options

The following fields in an i5/OS job definition map to options in the i5/OS SBMJOB command:

SBMJOB Options	Field in i5/OS Job definition
USER	User ID
JOBD	Job description
INLLIBL	Library list
JOBQ	Job queue
CURLIB	Current library
JOB	Job name

The special values for these SBMJOB options, such as *USRPRF and *JOBD, also apply to the i5/OS job definition fields. You can use these special values in your job definitions. For more information about the SBMJOB options and their special values, see the IBM documentation.

Set the Process Priority for an i5/OS Job

Process priority determines the order in which processes are scheduled on the System i5 processor. Depending on the priority level, process priority can speed up or slow down a process.

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

Follow these steps:

1. Open the i5/OS job that you want to set the priority level for.

   The Basic page opens in the i5/OS dialog.

2. Click Process Priority in the left pane.

   The Process Priority page opens in the right pane.

3. Select one of the following priorities in the Process priority field:

   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.

   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.

4. Click OK.

   The process priority for the i5/OS job is set.

Define an OpenVMS Job

You can define an OpenVMS job to schedule workload to run on an OpenVMS computer. The job runs an OpenVMS command file.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the OpenVMS job from the System group in the Palette view, and drag the job to the workspace.

   The OpenVMS icon appears on the Application workspace view.

3. Right-click the OpenVMS icon, and select Edit from the pop-up menu.

   The Basic page of the OpenVMS dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the OpenVMS agent where the job runs.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Command to run

   Specifies the path and command file name to execute when the job runs.

   Example: @SCRIPT:echo1.com

   Note: You can enter either an absolute path or just the logical name as long as you use valid syntax. The file extension and file version are optional but, if you specify the file version, you must also specify the file extension, for example, "RUN SAMPLE.EXE;25".

5. (Optional) Specify the following additional information:

   Arguments to pass

   Defines the argument string of positional parameters to pass to the command file. Arguments can be numeric or alphabetic strings of data.

   Note: You must specify each parameter in the order it is expected in the script, separating each parameter with a blank space. To pass a parameter containing spaces, enclose it in double quotation marks.

   Example: p1 "p7 p11"

   User ID

   Specifies the OpenVMS user ID that has the authority to run the job. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

   Examples: Bob, Production:Bob

   Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

6. Click OK.

   The OpenVMS job is defined.

Example: Run an OpenVMS Command

Suppose that an OpenVMS job OVMSJOB runs an OpenVMS command (@SCRIPT:echo1.com) with two arguments (p1, p4) under the PROD user ID. This job runs on the default agent computer.

To run an OpenVMS command

1. Enter the following information in the Basic page:
   - Name—OVMSJOB
   - Command to run—@SCRIPT:echo1.com
   - Arguments to pass—p1 p4
   - User ID—PROD
2. Click OK.

More Information:

Using a Namespace for a User that has Different Passwords (see page 63)

Environment Variables in OpenVMS

In OpenVMS, you use environment variables to define the local environment for the running of a command. You can add, update, and delete OpenVMS environment variables.

Program files can access environment variables using the getenv() function. Batch files can access environment variables by using the variable name and the trailing percent sign.

For example, you can set the MYDIR environment variable to alpha$dka200:[export]u1 in the Environment Variables page of the OpenVMS dialog.

More information:

Specify Environment Variables (see page 82)

Define a Tandem NSK Job

You can define a Tandem NSK job to schedule workload to run on a Tandem NSK computer. The job runs a Tandem NSK command file.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Tandem NSK job from the System group in the Palette view, and drag the job to the workspace.

   The Tandem NSK icon appears on the Application workspace view.

3. Right-click the Tandem NSK icon, and select Edit from the pop-up menu.

   The Basic page of the Tandem NSK dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the Tandem NSK agent where the job runs.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Command to run

Specifies the valid path and command file name to execute when this job runs. You can specify one of the following:

- The name of the file containing the macro (run \system.$*volume*.*subvolume*.*filename*)
- The name of the preloaded macro or routine (*macro_name*)
- The name of an obey file (obey *filename*)

5. (Optional) Specify the following additional information:

Arguments to pass

Defines the argument string of positional parameters to pass to the batch file or program. Arguments can be numeric or alphabetic strings of data.

Note: You must specify each parameter in the order it is expected in the script, separating each parameter with a blank space. To pass a parameter containing spaces, enclose it in double quotation marks.

Example: p1 "p7 p11"

User ID

Specifies the Tandem NSK user ID that has the authority to run the job. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

6. Click OK.

The Tandem NSK job is defined.

Example: Run a Tandem NSK Command

Suppose that a Tandem NSK job named TNSKJOB runs an NSK command (run \tag61.$data02.cyber.fastr) with two arguments (p7, p11) under the jdoe user ID. This job runs on the default agent computer.

To run a Tandem NSK command

1. Enter the following information in the Basic page:
   - Name—lTNSKJOB
   - Command to run—lrun \tag61.$data02.cyber.fastr
   - Arguments to pass—lp7 p11
   - User ID—ljdoe
2. Click OK.

Environment Variables in Tandem NSK

In Tandem NSK, the environment variables are generalized so that the system can execute any valid TACL command. You can use TACL variables to set up an appropriate environment on the Tandem NSK system before running the command.

To execute the TACL command, the name of the variable must be declared as TACLENV*x*, where *x* is the unique number in all TACLEV*x* variables. Each TACLENV*x* keyword must be unique but does not necessarily have to be entered in any particular sequence.

For example, you can set the following TACL variables in the Environment Variables page of the Tandem NSK dialog:

- taclenv1—load\tag661.$data02.cybtom.payrollcrm
- taclenv2—clear assign str derr

More information:

Specify Environment Variables (see page 82)

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.

Define a UNIX Job

You can define a UNIX 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the UNIX job from the System group in the Palette view, and drag the job to the workspace.

   The UNIX icon appears on the Application workspace view.

3. Right-click the UNIX icon, and select Edit from the pop-up menu.

   The Basic page of the UNIX dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent

   Specifies the name of the agent installed on the UNIX computer or the name of the agent group where the job runs. You can use agent groups for load balancing or to run a job on all agents in a group.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Specify action to take

   Indicates whether the job runs a script or executes a command.

   Note: You can also browse for the script or command name using the Script/Command Browser, enter the script or command name without the full path, or enter the script or command name using an environment variable.

5. (Optional) Specify the following additional information:

Arguments to pass

Defines the argument string of positional parameters to pass to a script. Arguments can be numeric or alphabetic strings of data.

Note: You must specify each parameter in the order it is expected in the script, separating each parameter with a blank space. To pass a parameter containing spaces, enclose it in double quotes.

Examples: p1 p2 p3, 362 "629 630" 748

User ID

Specifies the UNIX user ID that has the authority to run the job on the agent computer. You must specify a user ID if the job runs a command. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Default: Owner of the script or command (if the agent runs under the root account)

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- If the agent runs under another account, the job runs under the same account that runs the agent. The user ID or account must have the authority to run the script or command; otherwise, the job fails.

Shell

Indicates the shell name used to execute the script or command file. The default for most agents is the Korn shell (/bin/ksh). However, you can choose one of the following shells:

- /bin/csh—C shell
- /bin/bash—Bourne Again shell
- /bin/ksh—Korn shell
- /bin/sh—Bourne shell

6. Click OK.

The UNIX job is defined.

Example: Run a Script that is Located in a Path Set in the PATH Environment Variable

Suppose that the job PROCSCRIPT runs a script named procscript.sh. The job runs under the user ID jsmith, who has the authority to run the script. The path to procscript.sh is set in the PATH system environment variable for jsmith. This job runs on the default agent computer.

To run a script that is located in a path set in the PATH environment variable

1. Enter the following information in the Basic page:
   - Name—PROSCRIPT
   - Script/command name—procscript.sh
   - User ID—jsmith
2. Select the Run a script option button.
3. Click OK.

Example: Run a Script that is Located in a Path Set in a User Environment Variable

Suppose that the job MYSCRIPTJOB runs a script named myscript.sh. The job runs under the user ID jsmith, who 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. This job runs on the default agent computer.

To run a script that is located in a path set in a user environment variable

1. Enter the following information in the Basic page:
   - Name—MYSCRIPTJOB
   - Script/command name— $MY_PATH/myscript.sh
   - User ID—jsmith
2. Select the Run a script option button.
3. Click OK.

Example: Run a Different Script Depending on the Agent

Suppose that a job runs a payroll script (/export/home/cybuser/payroll.sh) on the UNIX1 agent. Sometimes, the job runs under a different agent. In this case, the job needs to run another script (export/home/cybuser/other_script.sh).

The %IF statement for the job's Script/command name field is as follows:

```
%IF(WOB._Agent=='UNIX1','/export/home/cybuser/payroll.sh','/export/home/cybuser/other_script.sh')
```

The WOB._Agent symbolic variable represents the agent that the job runs on.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Example: Run a PERL Script

Suppose that you want to schedule a PERL script located at /home/espuser/esptest/esptest.pl, and the PERL executable is located at /usr/contrib/bin/perl. The job PERLJOB runs under the user jsmith, who has access to the PERL executable and the authority to run the PERL script. This job runs on the default agent computer.

To run a PERL script

1. Enter the following information in the Basic page:
   - Name—PERLJOB
   - Script/command name—/usr/contrib/bin/perl
   - Arguments to pass—/home/espuser/esptest/esptest.pl
   - User ID—jsmith
2. Select the Execute a command option button.
3. Click OK.

Script and Command Name Specifications in a UNIX Job

To specify the script or command name in a UNIX job, you specify the path to and name of the script or command that the job runs.

You can also specify the script or command name in the following ways:

- You can browse for the path to and name of the script or command using the Script/Command Browser.

  Note: To use the Script/Command Browser, you must be connected to the server, and the agent defined for the job must be running.

- You can enter the script or the command 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.
  - The user ID you enter in the User ID field 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.
- You can enter the script or command name using an environment variable, such as $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.
  - The user ID you enter in the User ID field 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.

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 *ESP System Agent Administrator's Guide*.

How the Agent Selects the Shell to Use for a UNIX Script

To run a UNIX script, the agent selects the shell to use as follows:

1. The agent uses the Shell field if specified in the job definition.
2. The agent uses the first line of the script if the shell is not specified in the Shell field.
3. The agent uses the oscomponent.defaultshell parameter in the agent's agentparm.txt file if the shell is not specified in the Shell field or in the script.
4. The agent uses the user default shell defined in the user profile if not specified in one of the previous three locations.

Note: The agent administrator must define all of the shells that the agent uses in the agentparm.txt file, unless the oscomponent.checkvalidshell parameter is set to false. The shells are defined using the oscomponent.validshell parameter.

Set Process Priority for a UNIX Job

You can set the process priority for a UNIX job to determine 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.

Follow these steps:

1. Open the UNIX job you want to set the process priority for.

   The Basic page opens in the UNIX dialog.

2. Click Process Priority in the left pane.

   The Process Priority page opens in the right pane.

3. Select one of the following priorities in the Process priority field:

   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.

   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.

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.

4. Click OK.

   The process priority for the UNIX job is set.

Modify Resource Limits for a UNIX Job

When you define a UNIX job, you can set the job to modify 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.

Note: The resource limits 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.

Follow these steps:

1. Open the UNIX job you want to modify the resource limits for.

   The Basic page opens in the UNIX dialog.

2. Click User Limits in the left pane.

   The User Limits page opens in the right pane.

3. Select the resource type in the User Limits table. Options are the following:

   Maximum size of core files created

   Specifies the core file size in kilobytes (KB).

   Maximum size of a process's data segment

   Specifies the data segment size in kilobytes (KB).

   The maximum file size

   Specifies the maximum file size in kilobytes (KB).

   The process virtual size

   Specifies the process virtual size in kilobytes (KB).

   Maximum number of files

   Specifies the number of files.

   Maximum stack size

   Specifies the stack size in kilobytes (KB).

The maximum amount of CPU time

Specifies the CPU time in seconds.

4. Click Edit.

   The Edit Resource Usage Limits dialog opens.

5. Complete the following two fields:

   Soft Limit 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 digits; the value must be less than or equal to the hard limit

   Note: To specify an unlimited value for the resource, select the Unlimited check box.

   Hard Limit Value

   Defines the maximum usage (hard) limit for the specified resource type. The hard limit can only be increased by the root user. 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. Any user ID can decrease a hard limit.

   Limits: Any digits; the value must be greater than or equal to the soft limit

   Note: To specify an unlimited value for the resource, select the Unlimited check box.

6. Click OK.

   The specified soft and hard limits are shown in the User Limits table.

7. (Optional) Repeat Steps 3 to 6 to modify additional resource limits.

8. Click OK.

   The resource limits for the UNIX job are set.

Example: Specify Multiple Resource Limits

This example modifies the following resource limits on the UNIX computer for the duration of the job:

- The core file size limit is 100 KB (soft limit). The size can increase to 200 KB (hard limit).
- The process virtual size limit is 3332 KB (soft limit). The size can increase to an unlimited value.
- 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).

To specify multiple resource limits

1. Open the User Limits page.
2. Select the Maximum size of core files created resource in the User Limits table, and click Edit.

   The Edit Resource Usage Limits dialog opens.
3. Enter the following values, and click OK:
   - Soft Limit Value [KB]—100
   - Hard Limit Value [KB]—200

   The specified soft and hard limits are shown in the User Limits table.
4. Select the The process virtual size resource in the User Limits table, and click Edit.

   The Edit Resource Usage Limits dialog opens.
5. Enter **3332** in the Soft Limit Value [KB] field.
6. Select the Unlimited check box next to the Hard Limit Value [KB] field, and click OK.

   The specified soft and hard limits are shown in the User Limits table.
7. Select the Maximum stack size resource in the User Limits table, and click Edit.

   The Edit Resource Usage Limits dialog opens.
8. Enter the following values, and click OK:
   - Soft Limit Value [KB]—250
   - Hard Limit Value [KB]—300

   The specified soft and hard limits are shown in the User Limits table.
9. Select the The maximum amount of CPU time resource in the User Limits table, and click Edit.

   The Edit Resource Usage Limits dialog opens.
10. Enter the following values, and click OK:

- Soft Limit Value [seconds]—1000
- Hard Limit Value [seconds]—4000

The specified soft and hard limits are shown in the User Limits table.

11. Click OK.

Environment Variables in UNIX

In UNIX, you use environment variables to define the local environment to run a script. You can use UNIX environment variables or define your own. You can also add or update UNIX environment variables.

You can pass the following UNIX environment variables to a script:

HOME

Specifies the name of the user's initial working directory, and is used to find the .profile, .cshrc, and .login scripts.

Default: Home directory of the user running the script.

LANG

Specifies the name of the predefined setting for locale.

LC_ALL

Specifies the locale to be used if any of the five LC_ symbols (LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, and LC_TIME) are not defined.

LC_COLLATE

Specifies the name of the predefined setting for locale.

LC_CTYPE

Specifies the name of the locale for character classification.

LC_MONETARY

Specifies the name of the locale for money-related information.

LC_NUMERIC

Specifies the name of the locale for numeric editing.

LC_TIME

Specifies the name of the locale for date- and time-formatting information.

LOGNAME

Specifies the name of the user's login account.

PATH

Specifies the sequence of path prefixes used by execlp() and execvp() in locating programs to run.

PWD

Specifies the present working directory. The agent changes to this directory prior to executing the script.

TERM

Specifies the user's terminal type.

TZ

Specifies the time zone information.

Example: Variable Names and Their Values

The following table displays examples of variable names and their values:

Variable Name	Value
HOME	/user/user1
INPUT	/home/test
PWD	/usr/scripts/dailyrun

More information:

Specify Environment Variables (see page 82)

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).

Define a Wake on LAN Job

You can define a Wake on LAN 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Wake on LAN job from the System group in the Palette view, and drag the job to the workspace.

   The Wake on LAN icon appears on the Application workspace view.

3. Right-click the Wake on LAN job icon, and select Edit from the pop-up menu.

   The Basic page of the Wake on LAN dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that broadcasts the Wake on LAN (WOL) signal to a computer that has been soft-powered-down.

   Broadcast 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.

   Note: The agent must be on the same subnet as the computer it is trying to wake.

MAC address

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 (-) or colons (:). 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).

Example: 12-25-02-AF-2B-1A

5. (Optional) Specify the following additional information:

Password

Specifies the Wake on LAN (WOL) password as a set of four or six 2-digit hexadecimal values separated by dashes (-), colons (:), or periods (.).

Example: AA-BB-CC-DD-EE-FF (The password is hidden in the client.)

Notes:

- To use this field, the computer's network card must 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.
- The password is not encrypted.

6. (Optional) Specify the following ping parameters to confirm that the computer is awake:

Host to ping

Specifies the host name or IP address of the computer that receives the Wake on LAN (WOL) signal. You must specify a value for this field to enable the Timeout and Port(s) fields.

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.

Timeout

Specifies the timeout for the ping in seconds.

Limits: Up to 5 digits

Default: 120 seconds

Port(s)

Specifies the ports to ping after the agent sends the Wake on LAN (WOL) signal. To specify multiple ports, separate each port with a comma.

Default: 21, 22, 23, 80, 111, 135, 139, 445

Example: 1000,2003

Note: If at least one of the ports is available, the job completes successfully; otherwise, it fails.

7. Click OK.

   The Wake on LAN job is defined.

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.

To broadcast the WOL signal and complete immediately

1. Enter the following information in the Basic page:
   - Name—WAKE_DEFAULT
   - Agent name—AGENTNME
   - Broadcast address—172.16.0.0
   - MAC address—00-11-43-73-38-DC
2. Click OK.

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.

To broadcast the WOL signal and ping a port

1. Enter the following information in the Basic page:
   - Name—WAKE_PORT7
   - Agent name—AGENTNME
   - Broadcast address—172.16.0.0
   - MAC address—00-1E-4F-C1-0F-FE
2. Enter the following information in the Ping parameters section:
   - Host to ping—host
   - Port(s)—7
3. Click OK.

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.

To broadcast the WOL signal including a password

1. Enter the following information in the Basic page:
   - Name—WAKE_PASSWORD
   - Agent name—AGENTNME
   - Broadcast address—172.16.0.0
   - MAC address—11-22-33-44-55-66
   - Password—AA-BB-CC-DD-EE-FF (The password is hidden in the client.)
2. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)
Wake on LAN Jobs (see page 462)

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.

Define a Windows Job

You can define a Windows 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.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the Windows job from the System group in the Palette view, and drag the job to the workspace.

   The Windows icon appears on the Application workspace view.

3. Right-click the Windows icon, and select Edit from the pop-up menu.

   The Basic page of the Windows dialog opens.

4. Complete the following required fields as appropriate:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent

   Specifies the name of the agent installed on the Windows computer or the name of the agent group where the job runs. You can use agent groups for load balancing or to run a job on all agents in a group.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

Command to run

Specifies the path to and name of the Windows command file the job runs.

Note: You can also browse for the command file using the Command Browser or enter the command file name without the full path.

5. (Optional) Specify the following additional information:

Arguments to pass

Defines the argument string of positional parameters to pass to the command file. Arguments can be numeric or alphabetic strings of data.

Note: You must specify each parameter in the order it is expected in the command file, separating each parameter with a blank space. To pass a parameter containing spaces, enclose its value in double quotes.

Example: "c:\Pay Data\salary.dat" "c:\Pay Data\benefits.dat"

User ID

Specifies the Windows user ID that runs the job. You are restricted to how you can access data on remote computers. To access restricted remote resources, you can run the job under a user ID that has access to those resources. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- You can specify a user ID only when the agent is running as a Windows service under the Local System account.

Interactive job

Indicates whether the Windows job runs in interactive mode. Interactive mode lets you view and interact with jobs that invoke Windows Terminal Services or user interface processes. To run a Windows job in interactive mode, the agent must run as a service and the administrator user ID must be defined in the Topology and specified in the job definition.

Default: The job runs in batch mode

Notes:

- 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 setting in this field.
- By default, the agent uses the default Windows shell explorer.exe. The agent administrator can specify an alternative Windows shell for interactive jobs by setting the oscomponent.shell parameter in the agentparm.txt file.

6. Click OK.

   The Windows job is defined.

Example: Access a Windows Network Resource with Universal Naming Convention (UNC) and Share Names

CA Workload Automation DE and the agent support Windows UNC names and share 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. A share name is an alias for the path in which the resource exists.

Suppose that the path c:\WINNT\Profiles\Visitor\Desktop\ has the share name MyDesktop. The command notify.cmd is in that path on the CYBNT server, and is accessed by the UNC name and share name in the job JOBC, which runs on the agent WINAGENT.

To access a Windows network resource with UNC and share names

1. Enter the following information in the Basic page:
   - Name—JOBC
   - Agent name—WINAGENT
   - Command to run—\\CYBNT\MyDesktop\notify.cmd
2. Click OK.

Example: Use Administrative Authority to Access a Remote Windows Resource that is Not Shared

The agent service can log on to a remote Windows server as a user with administrative authority. The agent can then access remote resources using the share names C$ and ADMIN$, letting the agent access remote resources that are not marked as shared.

Suppose that drive C is accessed by an administrator over the network through the WINAGENT agent. The agent runs under the System Account option, and it runs the test Application in the c:\working directory on the CYBNT server. 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.

To use administrative authority to access a remote Windows resource that is not shared

1. Enter the following information in the Basic page:
   - Name—SHAREFILE
   - Agent name—WINAGENT
   - Command to run—\\CYBNT\C$\working\test
   - User ID—admin1
2. Click OK.

Example: Run cmd.exe

Suppose that you want the job COPYFILE to use the Windows command prompt (cmd.exe) to copy the file c:\env.txt to the test directory. To pass arguments to cmd.exe, you must enclose the argument in double quotes and precede the argument with the /C switch. The job runs on the default agent computer.

To run cmd.exe

1. Enter the following information in the Basic page:
   - Name—COPYFILE
   - Command to run—c:\Windows\system32\cmd.exe
   - Arguments to pass—"/C copy c:\env.txt c:\test\env.txt"
2. Click OK.

Example: Pass Arguments to a Visual Basic Script

Suppose that you want the job VBS to pass three arguments (one, two, and thee) to the Visual Basic script located at D:\temp\vbs\params.vbs. To pass arguments to a Visual Basic script, specify the location of the Visual Basic script and its arguments in the Arguments to pass field. This job runs on the default agent computer.

To pass arguments to a Visual Basic Script

1. Enter the following information in the Basic page:
   - Name—VBS
   - Command to run—c:\winnt\system32\wscript.exe
   - Arguments to pass—D:\temp\vbs\params.vbs one two three
2. Click OK.

Example: Run a Windows Job on a Different Server Than Usual

Suppose that a company performs scheduled maintenance on its NT_TOR1 agent on the last Sunday of every other month. On that day, work that usually runs on NT_TOR1 must run on NT_TOR2.

The following conditional logic uses the month number to determine if a month is an even-numbered month. The % (remainder) operator divides the value of day by 2 and tests the remainder. If the remainder is 0, the agent name value is set to NT_TOR2.

The %IF statement for the job's Agent name field is as follows:

```
%IF(today('last sunday of month') && APPL._AMM%2 == 0,'NT_TOR2','NT_TOR1')
```

The APPL._AMM symbolic variable represents the number of the actual month. If today is the last Sunday of an even month, the logical expressions returns true, and the job runs on NT_TOR2. If today is not the last Sunday of an even month, the logical expression returns false, and the jobs runs on NT_TOR1.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Example: Run a Different Command File Depending on the Agent

Suppose that a job usually runs a payroll command file (c:\Windows\Payroll\payroll.exe) on the NT_TOR1 agent. Sometimes, the job runs under a different agent. In this case, the job needs to run an alternative command file (c:\Windows\Payroll\other_command.exe').

The %IF statement for the job's Command to run field is as follows:

```
%IF(WOB._Agent=='NT_TOR1','c:\\Windows\\Payroll\\payroll.exe',
'c:\\Windows\\Payroll\\other_command.exe')
```

Note: JavaScript treats backslashes (\) as special characters, so you must precede each backslash in file paths with another backslash.

The WOB._Agent symbolic variable represents the agent the job runs on.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

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.

To run a Windows job in interactive mode

1. Enter the following information in the Basic page:
   - Name—EDIT_FILE
   - Agent name—WINAGENT
   - Command to run—notepad.exe
   - Arguments to pass—c:\run_info\config.txt
   - User ID—admin1
2. Select the Interactive job check box.
3. Click OK.

More information:

Using a Namespace for a User that has Different Passwords (see page 63)
Windows Jobs (see page 467)
Run a Job on an Agent Group for Load Balancing (see page 77)
Run a Job on All Agents Defined in a Group (see page 78)

Command File Specifications in a Windows Job

To specify the command file in a Windows job, you specify the path to and name of the command file that the job runs.

You can also specify the command file in the following ways:

- You can browse for the path to and name of the command file using the Command Browser.

  Note: To use the Command Browser, you must be connected to the server, and the agent defined for the job must be running.

- You can enter the command file without the full path if the agent is configured to search for paths to command files and the 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

Windows Job Objects

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 of the processes associated with a job object complete, the job object no longer exists.

More information:

Define a Windows Job Object (see page 475)

Define a Windows Job Object

You can define a Windows job object to manage the processing properties of various jobs as a single entity.

Follow these steps:

1. Open the Windows job you want to define the Windows job object for.
2. Click Job Object in the left pane.

   The Job Object page opens in the right pane.
3. Enter *one* of the following fields as appropriate:

   Create job object

   Defines the name of the new job object. This name must be unique.

   Assign job object

   Specifies the name of an existing job object to which you want to add this job.
4. (Optional) Specify the following additional information:

   Job memory

   Defines the maximum virtual memory in bytes allocated to *all* processes associated with the job object. If the total memory used for all processes associated with the job object exceeds this limit, the job that is trying to use memory fails.

   Note: To specify memory in megabytes, add M after the memory value. To specify memory in kilobytes, add K after the memory value.

   Examples: 50M, 500K

   Process memory

   Defines the maximum virtual memory in bytes allocated to *each* process associated with the job object. If the memory used for a single process exceeds this limit, the job fails.

   Note: To specify memory in megabytes, add M after the memory value. To specify memory in kilobytes, add K after the memory value.

   Examples: 50M, 500K

   Job time

   Defines the maximum CPU time in milliseconds allocated to *all* processes associated with the job object. If the total CPU time for all processes associated with the job object exceeds this limit, all jobs associated with the job object fail.

   Process time

   Defines the maximum CPU time in milliseconds allocated to *each* process associated with the job object. If the CPU time used for a single process exceeds this limit, the job fails.

Priority class

Indicates 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.
- 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.

Active process limit

Defines the maximum number of simultaneously active processes allowed in the job object.

Note: If you create a new job object and you do not specify a value for a job object property, that property has an unlimited value. If you assign an existing job object and you do not specify a value for a property, that property keeps the existing value.

5. Click OK.

   The Windows job object is defined.

Example: Define a Windows Job Object

Suppose that you want a Windows job to create a Windows job object named PayJobsObject. PayJobsObject can use a total of 40 MB of memory (41943040 bytes) and 1 hour of CPU time (4000000 milliseconds) for all processes it contains. Each process associated with PayJobsObject can use a maximum of 500 KB of memory (512000 bytes) and 3 minutes of CPU time (180000 milliseconds). PayJobsObject can have a maximum of 10 simultaneously active processes.

To define a Windows job object

1. Enter the following information in the Job Object page:
   - Create job object—PayJobsObject
   - Job memory—41943040
   - Process memory—512000
   - Job time—4000000
   - Process time—180000
   - Active process limit—10
2. Click OK.

More information:

Windows Job Objects (see page 474)

Environment Variables in Windows

In Windows, you use environment variables to define the local environment the command runs in. You can also add, update, and delete Windows environment variables.

Example: Variable Name and Values

The following table displays examples of variable names and their values:

Variable Name	Value
INPUT	c:\run.bat
NAME	Jane Doe
JOB	PAY
HOME	c:\export\u1

More information:

Specify Environment Variables (see page 82)

z/OS Jobs

You can use z/OS jobs to run mainframe workload.

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

CA WA Agent for z/OS submits and tracks the z/OS jobs. You can define the following three types of z/OS jobs:

z/OS-Data Set Trigger

Creates dependencies on data set activity.

z/OS-Manual

Creates dependencies on z/OS jobs that are submitted outside of the scheduling manager.

z/OS-Regular

Creates dependencies on z/OS jobs that are submitted outside of the scheduling manager.

z/OS-Data Set Trigger Jobs

You can define z/OS-Data Set Trigger jobs to create dependencies on data set activities. You can customize trigger conditions to define the circumstances in which the z/OS-Data Set Trigger job completes. After a z/OS-Data Set Trigger job completes, the server releases the successor jobs.

You can specify trigger conditions for the following data set activities:

- When a data set is created or updated
- When a specific job, group of jobs, or user ID creates a data set
- When an explicit data set notification is received (used when the data set activity does not generate an SMF record)
- When an FTP file is sent or received successfully

Note: Each data set must have its own individual Data Set Trigger job. To create dependencies on multiple data sets, you must create multiple Data Set Trigger jobs.

When specifying a data set, you can either specify the name of a single data set or use the hyphen wildcard to specify a group of data sets that match the selection criteria.

The following table shows how the hyphen wildcard matches a group of data sets:

Specified data set name	Matching data sets
CYB1.PAYROLL.A	CYB1.PAYROLL.A
CYB1.PAYROLL.G-	CYB1.PAYROLL.G0145V00 CYB1.PAYROLL.G0146V00 CYB1.PAYROLL.G0147V00 CYB1.PAYROLL.GRANTED.MORE
CYB1.P-	CYB1.PAYROLL CYB1.POSTJOBS CYB1.PROD.INPUT

Define a z/OS-Data Set Trigger Job

You can define a z/OS-Data Set Trigger job to create dependencies on data set activities.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the z/OS-Data Set Trigger job from the z/OS group in the Palette view, and drag the job to the workspace.

   The z/OS-Data Set Trigger icon appears on the Application workspace view.

3. Right-click the z/OS-Data Set Trigger icon, and select Edit from the pop-up menu.

   The Basic page of the z/OS-Data Set Trigger dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Agent name

   Specifies the name of the agent that monitors data set activity.

   Data set name

   Specifies the name of the data set that the agent monitors. You can either specify the name of a single data set or use the hyphen wildcard to specify a group of data sets that match the selection criteria.

   Examples: CYB1.PAYROLL.A, CYB1.PAYROLL.G-

   Note: You cannot use the asterisk (*) as a wildcard.

z/OS user ID

Specifies the z/OS user ID that owns the job on z/OS.

5. (Optional) Specify the following additional information:

Updated

Indicates whether to monitor for data set updates in addition to data set creation.

Default: Data set creation only

6. Click Trigger Conditions in the left pane.

The Trigger Conditions page opens in the right pane.

7. (Optional) Specify the following additional information:

Trigger when action is performed by

Indicates whether to restrict the trigger to data set activity performed by a specific job or user.

Name

Specifies the name of the job or the user who performs the data set activity.

Explicit data set trigger

Monitors for an explicit data set notification (used when the data set activity does not generate an SMF record).

FTP

Monitors for a successful FTP transfer.

Host

Specifies a host name to restrict the trigger to FTP transfers to or from a specific remote host.

Logon

Specifies a user ID to restrict the trigger to FTP transfers with a specific logon user ID.

Trigger after

Specifies the number of actions that must occur.

8. Click OK.

The z/OS-Data Set Trigger job is defined.

Example: Monitor for Data Set Creation and Update

Suppose that you want a z/OS-Data Set Trigger job named PROD.NIGHTLY to release its successors when the data set PROD.CICS.FILE1602 is closed (created or updated). The agent ZOS1 monitors the data set under user CYBDL01.

To monitor for data set creation and update

1. Enter the following information in the Basic page:
   - Name—PROD
   - Qualifier—NIGHTLY
   - Agent name—ZOS1
   - Data set name—PROD.CICS.FILE1602
   - z/OS user ID—CYBDL01
2. Select the Updated check box.
3. Click OK.

Example: Restrict the Trigger to Specific Data Sets Created by a Particular Job

Suppose that you want a z/OS-Data Set Trigger job named PROD.PAY_DATA to release its successors when job ABC creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-).The agent ZOS1 monitors the data set under user CYBDL01.

To restrict the trigger to specific data sets created by a particular job

1. Enter the following information in the Basic page:
   - Name—PROD
   - Qualifier—PAY_DATA
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-
   - z/OS user ID—CYBDL01
2. Do the following in the Trigger Conditions page:
   a. Select the Job name option button.
   b. Enter **ABC** in the Name field.
3. Click OK.

Example: Restrict the Trigger to Specific Data Sets Created by a Particular User

Suppose that you want the z/OS-Data Set Trigger job PROD.PAY_DATA to release its successors when the user CYBER1 creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set under user CYBDL01.

To restrict the trigger to specific data sets created by a particular user

1. Enter the following information in the Basic page:
   - Name—PROD
   - Qualifier—PAY_DATA
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-
   - z/OS user ID—CYBDL01
2. Do the following in the Trigger Conditions page:
   a. Select the User ID option button.
   b. Enter **CYBER1** in the Name field.
3. Click OK.

Example: Run a Compress Job After 100 Closures of a Data Set

Suppose that you want the z/OS-Data Set Trigger job PROD.PAY_DATA to release a compress job named COPYJCL.COMPRESS after every 100 closures of the data set CYBER.COPY.JCL. The agent ZOS1 monitors the data set under user CYBDL01.

To run a compress job after 100 closures of a data set

1. Enter the following information in the Basic page:
   - Name—COPYJCL
   - Qualifier—COMPRESS
   - Agent name—ZOS1
   - Data set name—CYBER.COPY.JCL
   - z/OS user ID—CYBDL01
2. Enter **100** in the Trigger after field in the Trigger Conditions page.
3. Click OK.
4. Define the compress job as a successor to the z/OS-Data Set Trigger job (COPYJCL.COMPRESS) in the Application.

Example: Monitor for Data Set Received from a Remote FTP Partner

Suppose that you want the z/OS-Data Set Trigger job PROD.PAY_DATA to release its successors when a file is successfully received from a remote FTP partner, creating generation data set USER1.PAYROLL (USER1.PAYROLL.G-).The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To monitor for data set received from a remote FTP partner

1. Enter the following information in the Basic page:
   - Name—PROD
   - Qualifier—PAY_DATA
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-
   - z/OS user ID—CYBDL01
2. Select Receive from the FTP drop-down list in the Trigger Conditions page.
3. Click OK.

Example: Monitor for Data Set Sent to a Remote FTP partner

Suppose that you want the z/OS-Data Set Trigger job CYBER.XFER to release its successors when data set CYBER.XFER.001 is successfully sent from the local mainframe partner to a remote FTP partner. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To monitor for data set sent to a remote FTP partner

1. Enter the following information in the Basic page:
   - Name—CYBER
   - Qualifier—XFER
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS user ID—CYBDL01
2. Select Send from the FTP drop-down list in the Trigger Conditions page.
3. Click OK.

Example: Restrict Triggering to a Specific Host

Suppose that you want the z/OS-Data Set Trigger job CYBER.XFER to release its successors when a remote FTP partner with IP address 172.16.0.0 successfully transfers a file creating the data set CYBER.XFER.001. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering to a specific host

1. Enter the following information in the Basic page:
   - Name—CYBER
   - Qualifier—XFER
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS user ID—CYBDL01
2. Do the following in the Trigger Conditions page:
   a. Select Send from the FTP drop-down list.
   b. Enter **172.16.0.0** in the Host field.
3. Click OK.

Example: Restrict Triggering to a Specific User ID

Suppose that you want the z/OS-Data Set Trigger job CYBER.XFER to release its successors when a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming the user ID prefix of the local FTP partner is CYB (CYB-). The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering to a specific user ID

1. Enter the following information in the Basic page:
   - Name—CYBER
   - Qualifier—XFER
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS user ID—CYBDL01
2. Do the following in the Trigger Conditions page:
   a. Select the User ID option button and enter **CYB-** in the Name field.
   b. Select Receive from the FTP drop-down list.
3. Click OK.

Example: Restrict Triggering to a Specific Logon ID

Suppose that you want the z/OS-Data Set Trigger job CYBER.XFER to release its successors when a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming that the remote FTP partner did log on to the FTP server with the CYBER005 user ID. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering to a specific Logon ID

1. Enter the following information in the Basic page:
   - Name—CYBER
   - Qualifier—XFER
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS user ID—CYBDL01
2. Do the following in the Trigger Conditions page:
   a. Select Receive from the FTP drop-down list.
   b. Enter **CYBER005** in the Logon field.
3. Click OK.

More information:

z/OS-Data Set Trigger Jobs (see page 478)

Define a z/OS-Manual Job

You can define a z/OS-Manual job to create dependencies on z/OS jobs that are submitted outside the scheduling manager, such as a job that is submitted manually by a user. After the manually submitted job completes, the server releases the successors of the Manual job.

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

If more than one generation of the Application is active when a manually submitted job completes, the server posts the Manual job as complete in all active Applications.

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the z/OS-Manual job from the z/OS group in the Palette view, and drag the job to the workspace.

   The z/OS-Manual icon appears on the Application workspace view.

3. Right-click the z/OS-Manual icon, and select Edit from the pop-up menu.

   The Basic page of the z/OS-Manual dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job submitted outside CA Workload Automation DE.

   Agent name

   Specifies the name of the agent that monitors the manually submitted job.

   z/OS user ID

   Specifies the z/OS user ID that owns the job on z/OS.

5. (Optional) Specify the following additional information:

   Authorization string

   Defines the authorization string the server uses to post the correct job. The server posts the Manual job complete when the manually submitted job runs under the corresponding user ID.

   Note: Contact your agent administrator to confirm whether the agent checks the authorization string. Depending on the scenario, the action to take differs as follows:

   - If authorization checking (AUTHSTR) is not set up for this agent, do not use this field.
   - If authorization checking (AUTHSTR) is set to RACUSER, enter the user ID that owns the job.
   - If authorization checking (AUTHSTR) is set to ACCOUNT1, ACCOUNT2, ACCOUNT3, or ACCOUNT4, enter the value found in the corresponding position of the job's JCL. For example, if AUTHSTR is set to check the ACCOUNT2 field, enter the value found in the second accounting position in the job's JCL.

   Search backward

   Defines the number of hours and minutes, in the format hhh:mm, that the agent does a backward search for a manually submitted job.

   Limits: 546 hours

   Example: 000:30

6. Click OK.

   The z/OS-Manual job is defined.

Example: Post z/OS Manual Job Complete Based on the User ID

Suppose that you want to post a z/OS Manual job complete when the manually submitted job ABC runs under CYBER. The ZOS1 agent monitors job ABC.

To post z/OS Manual job complete based on the user ID

1. Enter the following information in the Basic page:
   - Name—ABC
   - Agent name—ZOS1
   - Authorization string—CYBER
2. Click OK.

Define a z/OS-Regular Job

You can define a z/OS-Regular job to schedule a z/OS job.

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

Follow these steps:

1. Open the Application that you want to add the job to in the Define perspective.

   The Application appears in the workspace.

2. Select the z/OS-Regular job from the z/OS group in the Palette view, and drag the job to the workspace.

   The z/OS-Regular icon appears on the Application workspace view.

3. Right-click the z/OS-Regular icon, and select Edit from the pop-up menu.

   The Basic page of the z/OS-Regular dialog opens.

4. Complete the following required fields:

   Name

   Defines the name of the job that you want to schedule.

   Limits: 128 alphanumeric characters, plus the special characters commercial at (@), pound (#), dollar sign ($), underscore (_), square brackets ([]), brace brackets ({}), and percent sign (%) as a symbolic variable introducer character.

   Note: The name must match the name on the jobcard.

   Agent name

   Specifies the name of the agent where the mainframe workload runs.

   JCL Library

   Specifies the library name containing the JCL for this job.

   Example: cyber.prod.jcllib

   z/OS user ID

   Specifies the z/OS user ID that owns the job on z/OS.

5. (Optional) Specify the following additional information:

   JCL member

   Specifies the member name the JCL resides in for this job.

   Default: Job name

CopyJCL library

Specifies the copy JCL library that stores an editable working copy of the JCL you submitted.

Note: In case of a job failure, you can modify the working copy of the JCL and resubmit the job without affecting the JCL source. When you work with the JCL copy, the server automatically resolves all the variables.

6. Click OK.

   The z/OS-Regular job is defined.

Example: Store a Working Copy of the JCL that You Submitted

Suppose that the agent ZOS1 submits the JCL in member CYBDL01A in the CYBDL01.JCLLIB library. If the job fails, you can modify a working copy of the JCL in the CYBDL01.COPY.JCLLIB data set, and resubmit the job without affecting the JCL source.

To store a working copy of the JCL that you submitted

1. Enter the following information in the Basic page:
   - Name—CYBDL01A
   - Agent name—ZOS1
   - JCL library—CYBDL01.JCLLIB
   - z/OS user ID—CYBDL01
   - CopyJCL library—CYBDL01.COPY.JCLLIB
2. Click OK.

Condition Codes

Within the z/OS-Regular job definition dialog, you can use the Condition Codes page to specify the codes that indicate a successful or failed job.

You can define condition codes for jobs, steps, procedure steps, or programs as follows:

- A condition code for one number or a range of numbers between 1 and 4095 (ranges are separated by colons, such as 1:4095).
- A system abend code (S*ccc*), such as S0C1 or SB37.
- A user abend code (U*nnnn*), such as U0001 or U0462. The *nnnn* must be exactly four decimal digits and cannot exceed 4095.

For each condition code or range of condition codes, you can indicate whether the server is to interpret the code or range as a success or a failure. You can also indicate whether the job should continue or stop running. Each specification becomes a separate item. Together these specifications form a list of condition codes.

Example: Complete a Job Based on Specific Return Codes

Suppose that you define condition codes so that a job completes successfully only when the return code is 0, 1, 2, 3, 4, or 999.

To complete a job based on specific return codes

1. Open the z/OS-Regular job for which you want to specify condition codes.
2. Click Condition Codes in the left pane.

   The Condition Codes page opens in the right pane.
3. Click Add.

   A new row is added to the Condition code list.
4. Enter **1:4** in the Return Code field, select SUCCESS in the Interpret As drop-down list, and select CONTINUE in the Action Type drop-down list.
5. Click Add.

   A new row is added to the Condition code list.
6. Enter **999** in the Return Code field, select SUCCESS in the Interpret As drop-down list, and select CONTINUE in the Action Type drop-down list.
7. Click Add.

   A new row is added to the Condition code list.
8. Enter **5:998** in the Return Code field, select FAILURE in the Interpret As drop-down list, and select STOP in the Action Type drop-down list.
9. Click OK.

Enable CA WA Restart Option to Restart z/OS Jobs

You can enable CA WA Restart Option to restart z/OS jobs.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.
3. Click Restart Option EE in the left pane.

   The Restart Option EE page opens in the right pane.
4. Select the Include restart step check box.

   You can restart the job using CA WA Restart Option.

Specify CA WA Restart Option Statements

You can specify predefined and custom CA WA Restart Option statements to identify special processing options.

For example, you can instruct CA WA Restart Option to do the following:

- Back out data sets created by the job.
- Perform data set cleanup.
- Run the job despite errors predicted by CA WA Restart Option.
- Check for errors prior to a job's submission.
- Perform automatic recovery of missing data sets.
- Honor previously-coded condition codes and bypass steps accordingly.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click Restart Option EE in the left pane.

   The Restart Option EE page opens in the right pane.

4. Select the predefined statements you want CA WA Restart Option to send and the parameter to pass with each.

   Note: These statements depend on the defaults set in the initialization parameters on the CA WA Agent for z/OS. For more information about the initialization parameters, contact your agent administrator.

5. (Optional) Enter your own CA WA Restart Option statements starting at line 7.

   Note: For each statement, select the line in the Send column and enter the statement. Do not prefix the statement with ENCPARM. You can include up to 12 statements in total.

6. Click OK.

   The CA WA Restart Option statements are specified for the z/OS-Regular jobs in the Application.

Override the CA WA Restart Option Statements in the Job Definition

You can override the CA WA Restart Option statements specified in the Application in the job definition.

Follow these steps:

1. Open the z/OS-Regular job for which you want to specify condition codes.
2. Click Restart Option EE in the left pane.

   The Restart Option EE page opens in the right pane.
3. Clear the Use defaults check box.
4. Specify the CA WA Restart Option statements to send as appropriate.
5. Click OK.

   The CA WA Restart Option statements are overridden for this z/OS-Regular job.

Chapter 3: Working with Events

This section contains the following topics:

Events (see page 493)
Create an Event (see page 505)
Locate a Job in the Graphical Simulation (see page 508)
Specify Suspend and Resume Times (see page 509)
Prevent an Event from Triggering If an Application is Active (see page 511)
Schedule Workload (see page 512)
Specify Application Parameter Values in a Date-Time/Manual Event (see page 514)
Remove Application Parameters from a Date-Time/Manual Event (see page 515)
Trigger Workload When a File is Created (see page 516)
Trigger Workload Based on z/OS Data Set Activity (see page 519)
Trigger Workload When a JMS Message is Received (see page 526)
Trigger Workload When an SAP Event is Raised (see page 529)
Trigger Workload When the Number of Rows in a Database Table Changes (see page 532)
Trigger Workload When a Row is Added, Deleted, or Updated in a Database Table (see page 535)
Trigger Workload When a Global Variable Expression is Met (see page 542)

Events

You can use Events to run the workload defined in your Applications. When an Event is triggered, the Application runs.

Events can be scheduled, triggered manually, or triggered based on a particular condition.

If the Event is scheduled, the Application runs based on the Event's schedule. For example, you can schedule an Event to run an Application daily at 10 a.m. You can also trigger the Event manually if you want to add a new scheduled Event execution to the schedule or replace the Event's next execution.

If the Event is not scheduled, it can be triggered manually or triggered based on a particular condition that an agent monitors. Monitor Events are continuous; the agent continues to monitor after the condition you specify occurs. For example, you can continuously monitor file activity and trigger workload each time a file is created.

An Application may contain multiple Events. For example, an Application may need to run each day at 4 p.m. and also whenever a specific file is created, a JMS message is received, or an SAP event is raised.

Event Triggers

You can specify the following types of triggers for an Event:

Date-Time/Manual

Triggers workload based on schedule criteria. For example, you can schedule an Event to run an Application daily at 10 a.m. If the Event does not have schedule criteria, the Event must be triggered manually.

Database Monitor

Triggers workload when the number of rows that satisfy a monitor condition increases or decreases in a database table. Database Monitor Events monitor for changes using a polling interval (every 10 seconds by default).

Database Trigger

Triggers workload when a row that satisfies a trigger condition is added, deleted, or updated in a database table. A Database Trigger Event creates a database trigger on the database.

File Trigger

Triggers workload based on file activity. You can only trigger on file creation.

JMS Subscribe

Triggers workload when a JMS message is received.

SAP Event Monitor

Triggers workload when an SAP Event is raised on the SAP system.

Variable Dependency Monitor

Triggers workload when a global variable expression is met.

z/OS Data Set Trigger

Triggers workload based on z/OS data set activity. You can trigger the Event on the creation, update, and FTP transfer of data sets.

More information:

Create an Event (see page 505)

Event Schedule Criteria

To schedule workload, you must create an Event and specify schedule criteria. The server triggers the Event and runs the Application at the scheduled dates and times.

The schedule criteria for an Event generally includes a time and a frequency.

You can specify an AM or PM time (such as 3:00 PM) or a 24-hour time (such as 15:00). You can also specify a time zone (such as 8PM ETZ).

If you specify a time without a date (such as 9PM), the server schedules the Application daily at that time.

If you specify a date without a time (such as 1ST MONDAY OF MONTH), the server resolves the time to midnight (00:00:00).

Example: Trigger an Event Daily

The following schedule criteria triggers an Event at 10 a.m. every day:

```
Schedule 10AM DAILY
```

Example: Trigger an Event at Different Times During the Week

The following schedule criteria triggers an Event at two different times during the week:

```
Schedule 4PM WEEKDAYS
Schedule 2PM WEEKENDS
```

Example: Specify Exceptions to the Schedule Criteria

Suppose that you want to schedule your Event at 5 p.m. every day, except on the last workday of the month. The schedule criteria is specified as follows:

```
Schedule 5PM DAILY
Do not schedule 5PM LAST WORKDAY OF MONTH
```

Note: You should specify Do not schedule statements after the Schedule statements. The Do not schedule time must match the time used in a Schedule statement (5PM in this example).

Example: Specify Exception to the Schedule Criteria on a Specific Date

Suppose that you want to schedule your Event every day except on August 23, 2006. The schedule criteria is specified as follows:

```
Schedule DAILY
Do not schedule AUGUST 23, 2006
```

More information:

Schedule Workload (see page 512)

Expect Times

For Events that are not scheduled, you can specify an expect time when you create the Event. The expect time indicates an expected time for the triggering of the Event. If an Event has an expect time, the Event appears on the schedule and forecasts for that time.

For example, suppose that a file is created around 4 p.m. each day. When you create a File Trigger Event that monitors the creation of the file, you can specify an expect time of 4PM DAILY. The File Trigger Event appears on the schedule for 4 p.m. each day.

Note: No action is required if the Event is not triggered by the expected time.

Workload Simulation

You should simulate an Event before it triggers and runs an Application. Simulating an Event displays the following:

- Which jobs would run if the Event was triggered on a certain day and the order the jobs would run in
- The details of the job definitions
- If the Application contains any JavaScript errors
- Resolution of symbolic variables that are resolved at trigger time

Note: If you simulate an Application that includes a JavaScript script that executes at run time, the simulation may fail because the server resolves some symbolic variables at run time, not at trigger time. You can ignore any simulation errors about unresolved symbolic variables. When the Event triggers the Application, the server resolves the symbolic variables at run time.

For scheduled Events, you can simulate the next execution, a future execution, or a past execution of an Event. You can also simulate an Event and specify a subset of jobs, known as root jobs, within the Application. If you do not specify a schedule criteria for a simulation, the server uses the Event's next scheduled time.

For non-scheduled Events, if you do not specify a schedule criteria for a simulation, the server uses the next expected time (if the Event has an expect time) or the current time.

Note: In simulations, you cannot specify multiple occurrences, such as DAILY, MONDAY WEDNESDAY FRIDAY, or 3RD-6TH DAY OF MONTH.

Suspend and Resume Times

Suspending an Event prevents the Event from triggering until you resume it.

Using Event-control criteria, you can schedule the suspending and resuming of Events. You might schedule an Event to occur within a range of time, resume an Event that was suspended manually, or suspend an Event at some time in the future.

Suspending a scheduled Event for some time in the future is useful if you are running an Application multiple times each day. Suspending a monitor Event for some time in the future is useful if you want to monitor for a trigger condition during a specific window of time.

Note: When you specify a suspend time, specify a time before the Event's scheduled or expected time; otherwise, the server may not suspend the Event in time. We recommend you use a suspend time of at least one minute before the Event's scheduled or expected time. For example, if your Event is scheduled for 8:00, to suspend it, specify a time of 7:59 or earlier. Similarly, to resume an Event, specify a resume time of at least one minute before you want the Event to resume.

When specifying suspend and resume times, consider the following:

- If you specify a day and month in your Event-control criteria, you must include the year, for example, APR 1 2006.
- If you want to suspend an Event on a particular date, specify an absolute date (for example, 9:01 March 1, 2006) instead of a relative date (for example, 9:01 TOMORROW).
- You cannot specify some scheduling terms in suspend and resume times, such as the following:
  - Starting time/date and ending time/date (such as HOURLY STARTING 2PM TODAY)
  - Every *n units* (such as, EVERY 5 MINUTES)
- When an Event is manually suspended, its suspend count increments by one. If you specify a scheduled resume using Event-control criteria, the suspend count decreases to zero regardless of its current suspend count.

More information:

Specify Suspend and Resume Times (see page 509)

Event Priority

In an Event definition, you can specify an Event priority to prioritize the triggering of Events scheduled at the same time. The Event priority is a number from 1 (lowest priority) through 10 (highest priority). The default Event priority is 5.

Whenever multiple Events are eligible for execution at the same time, the server observes the following rules:

- Events are triggered in descending order of their priority. The Event with the highest priority is triggered first.
- If multiple Events have the same priority, the server triggers the Event with the earliest execution time first.
- If multiple Events have the same priority and execution time, the server triggers the Event with the earliest trigger request time first.
  - For scheduled Event executions, the trigger request time is the time the Event was last updated or created.
  - For Event executions triggered manually, the trigger request time is the time the command was issued.
  - For Event executions triggered based on a particular condition, the trigger request time is the time the server processed the agent notification indicating the occurrence of the condition.

Example: Processing Multiple Events at the Same Time

If the server is processing multiple Events that trigger at the same time, an Event with a higher priority can execute before an Event that has an earlier execution time. Consider the following scenario:

- 10 Events with priority 5 are scheduled to trigger at midnight (00:00:00).
- An Event with priority 6 is scheduled to trigger one second later (00:00:01).

Assuming that the server takes more than one second to trigger the 10 Events, the Event scheduled at 00:00:01 may trigger before some of the Events scheduled at 00:00:00.

More information:

Create an Event (see page 505)

Date-Time/Manual Events

Date-Time/Manual Events let you schedule workload or trigger workload manually.

When you define a new Application in the Define perspective, CA WA Desktop Client defines a default Date-Time/Manual Event. By default, CA WA Desktop Client uses your user name as the Event prefix and your Application name as the Event name. You can use this Event to schedule the Application or to run the Application manually. You can also create additional Events in the Application.

With Date-Time/Manual Events, you use schedule criteria to define when the server triggers the Event. The schedule criteria for an Event generally includes a time and a frequency. For example, you can schedule an Event to run daily at 10 a.m.

You can also specify exceptions to the schedule criteria. For example, you can schedule an Event to run daily at 5 p.m. except on the last workday of the month.

If you do not specify schedule criteria in the Event, you must trigger the Event manually to run the Application. For example, suppose that you want to trigger a one-time Event to run the Application on a particular day. You would define the Event without schedule criteria and trigger it manually on that day.

Application Parameters

You can specify parameters and parameter values in Date-Time/Manual Event definitions. These values are used to resolve the corresponding parameters in the Application definition.

Note: You cannot overwrite parameter values at Event trigger time. You can specify them only at the time of Event definition.

The following considerations apply when specifying parameters and their values in Event definitions:

- The parameter list in the Event definition must match the parameters specified in the Application definition. CA WA Desktop Client does not validate that the Event parameters match the Application parameters.
- If you use parameters in required fields such as Agent name, you must provide values for those parameters. Otherwise, the Applications are not triggered. The same restriction applies to some of the optional fields including Qualifier. CA WA Desktop Client does not validate that all parameters have values.
- You can create Events for Applications that have not been created yet. You can modify the Application and Event definitions independent of each other.

More information:

Remove Application Parameters from a Date-Time/Manual Event (see page 515)
Specify Application Parameter Values in a Date-Time/Manual Event (see page 514)

File Trigger Events

You can continuously trigger workload based on file activity. For example, a File Trigger Event can monitor the creation of a file. Whenever the file is created, the server triggers the Event, which runs an Application to process the file. If the file being monitored is updated after a previous trigger, the Event will be triggered again.

Note: To use File Trigger Events, your system requires CA WA Agent for UNIX, Linux, or Windows.

Some usage scenarios of Event-level file triggers include the following:

- A file is created occasionally. Whenever the file is created, an Application needs to run to process the file.
- Transactions are received continuously throughout the day in the form of files written to a series of directories. When a file is received, a job must be submitted. The filename determines the name and executable of the job.
- An Application runs each day at 4 p.m. and also whenever a specific file is created.

File Trigger Events monitor file activity using a polling interval, which is every 30 seconds by default. File Trigger Events do not detect changes that cancel each other out during the polling interval. For example, if the file is created and deleted during the polling interval, the trigger does not occur. Because the file did not exist when the directory was polled, the Event does not detect the file creation and deletion.

Note: The agent administrator can change the number of seconds between polls by configuring the filemonplugin.sleepperiod parameter in the agentparm.txt file. For more information about the filemonplugin.sleepperiod parameter, see the agent *Implementation Guide* for your operating system.

Example: Comparison of Event-Level and Job-Level File Triggers

Suppose that an Application requires a file to be created before it can run each day. If the file is not created by a cutoff time, the Application should not run. You can use the following approaches for the workload:

- You can schedule an Event to run the Application each day. In the Application, a File Trigger job monitors the creation of the file and is marked overdue at the cutoff time. If the File Trigger job does not complete by the overdue time, a script runs to automatically complete the Application.
- You can use an Event to monitor the creation of the file. This approach is better because the Application builds only when the file is created, and you do not need to code a script.

Note: You can trigger workload based on file creation only. For other scenarios, you can use the Date-Time/Manual Event trigger with the File Trigger workload object.

More information:

Trigger Workload When a File is Created (see page 516)

z/OS Data Set Trigger Events

You can continuously trigger workload based on data set activity. For example, a z/OS Data Set Trigger Event can monitor the creation of a data set. Whenever the data set is created, the server triggers the Event, which runs an Application to process the data set.

Note: To use z/OS Data Set Trigger Events, your system requires CA WA Agent for z/OS.

You can specify Event trigger conditions for the following data set activities:

- When a data set is created or updated
- When a specific job, group of jobs, or user ID creates a data set
- When an explicit data set notification is received (used when the data set activity does not generate an SMF record)
- When a file is sent or received successfully by FTP

When specifying a data set, you can either specify the name of a single data set or use the hyphen wildcard to specify a group of data sets that match the selection criteria.

The following table shows how the hyphen wildcard matches a group of data sets:

Specified data set name	Matching data sets
CYB1.PAYROLL.A	CYB1.PAYROLL.A

Specified data set name	Matching data sets
CYB1.PAYROLL.G-	CYB1.PAYROLL.G0145V00 CYB1.PAYROLL.G0146V00 CYB1.PAYROLL.G0147V00 CYB1.PAYROLL.GRANTED.MORE
CYB1.P-	CYB1.PAYROLL CYB1.POSTJOBS CYB1.PROD.INPUT

More information:

Trigger Workload Based on z/OS Data Set Activity (see page 519)

JMS Subscribe Events

With JMS Subscribe Events, you can trigger workload whenever a JMS message is received (consumed) from a topic or a queue. You can selectively filter all messages against a text string you construct using regular expression logic. When the agent receives a message matching the filter criteria, the server runs the Application referenced in the Event. The agent sends the messages that meet the filter criteria to a destination file you specify.

Notes:

- To use JMS Subscribe Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.
- You can only use one JMS Subscribe Event for each queue.

More information:

Trigger Workload When a JMS Message is Received (see page 526)

SAP Event Monitor Events

With SAP Event Monitor Events, you can trigger workload whenever an SAP event is raised. In SAP, the term “raising an Event” is analogous to triggering an Event in CA Workload Automation DE. When the SAP event is raised on the SAP system, the server runs the Application referenced in the SAP Event Monitor Event.

Note: To use SAP Event Monitor Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

More information:

Trigger Workload When an SAP Event is Raised (see page 529)

Database Monitor and Database Trigger Events

You can continuously trigger workload based on database activity using Database Monitor or Database Trigger Events.

Note: To use Database Monitor and Trigger Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

A Database Monitor Event 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 Event definition. Whenever the condition is met, the server runs the Application referenced in the Event.

Database Monitor Events count the number of rows that satisfy the monitor condition using a polling interval, which is every 10 seconds by default. Database Monitor Events 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 Event does not detect the row addition and deletion.

A Database Trigger Event 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 Event definition. Whenever the condition is met, the server runs the Application referenced in the Event.

Each Database Trigger Event 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*DB_type_name*.properties, for example, dbtrigOracle.properties. Contact your database administrator before choosing to use a Database Trigger Event.

Note: A table that is being monitored should not be dropped. The Database Monitor or Database Trigger Event will remain active, even though the table has been dropped.

More information:

Trigger Workload When the Number of Rows in a Database Table Changes (see page 532)
Trigger Workload When a Row is Added, Deleted, or Updated in a Database Table (see page 535)

Variable Dependency Monitor Events

Variable Dependency Monitor Events let you trigger workload based on global variable expressions. When you create a Variable Dependency Monitor Event, the server evaluates the variable expression, and runs the Application referenced in the Event whenever the variable expression is satisfied. The server re-evaluates the expression whenever any of the variables in the expression changes.

You can construct variable expressions using the available global variables or by defining your own global variables.

For example, a weather office can trigger relaying a hurricane warning to multiple places if the humidity falls below a certain level and the wind speed crosses a limit. Similarly, a bank can trigger loan processing based on the score card and net asset value of applicants.

More information:

Trigger Workload When a Global Variable Expression is Met (see page 542)

Create an Event

To run the workload defined by an Application, you create an Event.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select the Event you want to create from the pop-up menu.

   The Event appears in the list of Events in the Application Events view.

   Note: You can limit which Event types appear on the toolbar to the left of the Events table area. For example, if you do not use SAP Event monitoring, from the main menu, select Window, Preferences, Desktop Client, Define Perspective, Event Types, clear the SAP Event Monitor check box, and click OK.

3. Right-click the Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Click Properties in the left pane.

   The Properties page opens in the right pane.

5. Complete the following fields as appropriate:

Event prefix

Defines a prefix for the Event.

Default: Name of the user defining the Event

Limits: Up to 32 characters

Note: To set a default prefix, from the main menu click Window, Preferences, Desktop Client, Services Perspective, Event, and enter the default prefix in the Prefix field.

Event name

Defines the name of the Event.

Default: A variation of the Application name

Limits: Up to 128 characters

Note: The prefix and descriptive name uniquely identify the Event on the system where it resides.

Execution user

(Optional) Specifies the execution user of the Event. When the Event is triggered, the Application triggered by the Event runs with the security permissions of the execution user.

Default: Name of the user defining the Event

Notes:

- When the Event triggers, the execution user requires the appropriate permissions to read the Application triggered by the Event.
- When each job runs in the Application, the execution user requires the appropriate permissions to run the job on the agent and to read any global variables specified in the job.

Inherit trigger user

Overrides the execution user for Events triggered manually. If this option is set, the execution user is set to the user that manually triggered the Event.

Note: To apply this option to all the Events you define, from the main menu click Window, Preferences, Desktop Client, Services Perspective, Event, and select the Inherit trigger user check box.

Priority

Specifies the Event priority as an integer from 1 (lowest priority) through 10 (highest priority). When multiple Events are eligible for execution at the same time, the server triggers the Events in descending priority order.

Default: 5

Note: To set a default Event priority, from the main menu click Window, Preferences, Desktop Client, Services Perspective, Event, and enter the default priority in the Priority field.

Specify Calendars

(Optional) Specifies up to two calendars that are used to schedule the Event.

Default: SYSTEM calendar

Note: If your Event or Application uses any terms in a calendar other than the SYSTEM calendar, specify those calendars in this section. You can set default calendars by selecting Window, Preferences, Desktop Client, Services Perspective, Event from the main menu.

6. Complete the remaining fields as appropriate, and click OK.
7. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   If it contains no errors, the Event is saved.

More information:

Event Triggers (see page 494)
Event Priority (see page 498)

Locate a Job in the Graphical Simulation

You can locate a specific job in the graphical simulation of the Application. This feature is a useful tool for simulating large Applications with many jobs.

Follow these steps:

1. Simulate an Application.

   A graphical and text-based representation of the Application appears.

2. Right-click in the Simulation workspace, and select Locate Job in Graphical Simulation from the pop-up menu.

   The Locate Job dialog opens.

3. Enter the job name in the Job name field, and click OK.

   Note: You must include a wildcard for a partial name. For example, A* locates all jobs with names that start with A.

   A list of all of the jobs that match the criteria appear in the list box.

   Note: If one job matches the criteria, the job is highlighted in the graph and you can omit the next step.

4. Select the job in the list box, and click Search.

   The job is highlighted in the Simulation workspace.

Specify Suspend and Resume Times

You can specify suspend and resume times to control when an Event can be triggered. For example, you can restrict the triggering of an Event within a specific window of time.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Event in the Application Events view, and select Edit from the pop-up menu.

   The Event definition dialog opens.

3. Click Schedule in the left pane.

   The Schedule page opens in the right pane.

4. Complete the following fields as appropriate:

   Suspend at

   Specifies a date and time to suspend the Event.

   Note: To select a date and time, you can click the button to the right of the Suspend at field.

   Resume at

   Specifies a date and time to resume the Event.

   Note: To select a date and time, you can click the button to the right of the Resume at field.

5. Click OK.

   The suspend and resume times are specified.

6. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Event is saved if it contains no errors.

Note: To ensure that you entered the correct Event-control criteria, you can use the List Scheduled Events feature to test it.

Example: Suspend an Event at a Future Date

Suppose that today's date is March 15, 2006, and you want to prevent an Event from triggering starting on April 1, 2006.

To suspend an Event at a future date, enter **apr 1 2006** in the Suspend at field of the Schedule page.

On April 1, 2006, the server suspends the Event, preventing the Event from triggering. When you want the Event to resume its execution, resume the Event.

Example: Monitor File Creation Within a Specific Window of Time

Suppose that an Application requires an input file each day before it runs. The cutoff time for this file is 7 p.m. If the input file is not received by this cutoff time, the Application should not run.

To monitor file creation within a specific window of time

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Define a File Trigger Event to monitor the creation of the input file.
3. Click Schedule in the left pane of the Event definition dialog.

   The Schedule page opens in the right pane.
4. Enter **19:00 daily** in the Suspend at field in the Specify Event control criteria section.
5. Enter **00:00 daily** in the Resume at field in the Specify Event control criteria section.
6. Click OK.
7. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   At 7 p.m. each day, the server suspends the Event until the beginning of the next day. If the input file is received after 7 p.m., the Application does not run. If the input file is received before 7 p.m., the server triggers the Event that runs the Application to process the file.

More information:

Suspend and Resume Times (see page 497)

Prevent an Event from Triggering If an Application is Active

You can prevent Application generations from running concurrently by preventing an Event associated with the Application from triggering while the current generation (triggered by that same Event) is running.

Note: An Application can be triggered by multiple Events, each with its own scheduling criteria. If you want to prevent Application generations from running regardless of which Event triggered the Application, you can use a similar option at the Application level.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Event in the Application Events view, and select Edit from the pop-up menu.

   The Event definition dialog opens.

3. Click Schedule in the left pane.

   The Schedule page opens in the right pane.

4. Select the Do not trigger if active check box, and click OK.

5. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   If a generation of the Application is running and the next scheduled execution of the Event occurs, the server does not trigger the Event, causing the Event to miss its scheduled execution time. The Event triggers at its next scheduled time when the current generation completes.

Note: If the server is restarted, Events you defined with the Do not trigger if active option may be suspended. Following a cold start, check the state of these Events and manually resume Events that the server may have suspended. For more information about cold start, see the *Implementation Guide*.

More information:

Concurrent Application Generations (see page 18)

Schedule Workload

To schedule workload, you define a Date-Time/Manual Event and specify schedule criteria.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Data-Time/Manual Event in the Application Events view, and select Edit from the pop-up menu.

   Note: If the Date-Time/Manual Event does not appear in the Application Events view, right-click within the Events table area, and select New Date-Time/Manual from the pop-up menu.

   The Event definition dialog opens.

3. Click Schedule in the left pane.

   The Schedule page opens in the right pane.

4. Click Add Schedule in the Specify schedule criteria section.

   A new row is added in the Specify schedule criteria table.

5. Click the ellipses (…) in the When field to build and test your schedule criteria.

   Note: If you are familiar with the scheduling terms, you can type your schedule criteria, such as 3AM, HOURLY, or 10AM DAILY, directly in the When field.

   The Schedule event dialog opens.

6. Select either the Time check box in the Time/Time Zone section and specify a time in the text box or select the Every check box in the Interval section and specify the interval in the text boxes.

   Note: If you select the Time check box, you can select a frequency from the Frequency section and make adjustments to your schedule criteria for holidays in the On holiday section. If you select the Every check box, you can specify intervals such as hourly, every 2 days, every week, and so on.

7. Click Test to test your schedule criteria.

   The Test Results text box displays the date and time of the next 10 Event executions.

8. Click OK to accept your scheduling criteria.

   The Schedule event dialog closes and the When field displays your schedule criteria.

9. (Optional) Click Add Schedule to add another schedule statement or click Add Do not schedule to specify an exception to the schedule criteria.

10. Complete the remaining fields as appropriate, and click OK.

11. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

    The Event is saved if it contains no errors. The server triggers the Event and runs the Application at the scheduled dates and times.

Example: Specify Multiple Schedule Criteria

For complicated schedule criteria, you can specify multiple schedule statements.

The following schedule criteria schedules an Event at 9:15, 11:15, and 13:15 every day:

```
Schedule DAILY AT 09:15
Schedule DAILY AT 11:15
Schedule DAILY AT 13:15
```

Note: If you enter the above statements in a single line (DAILY AT 09:15 11:15 13:15), the server schedules the Event only once, at 13:15.

Examples: Specify a Start and End Times

You can specify a start time and date, an end time and date, or both.

The following schedule criteria triggers an Event hourly starting at 2 p.m. today:

```
Schedule HOURLY STARTING 2PM TODAY
```

The following scheduling criteria stops triggering an Event on July 1, 2006:

```
Schedule DAILY ENDING 1JUL2006
```

The following schedule criteria triggers an Event at 4 p.m. daily starting on November 6, 2006 and ending at 4:01 p.m. on April 22, 2007:

```
Schedule 4PM DAILY STARTING NOV 6,2006 ENDING 4:01PM APR 22,2007
```

Example: Specify a Recurring Event

You can schedule a recurring Event for every *n units* of time where *units* can be seconds, minutes, hours, days, workdays, weekdays, weeks, months, or years.

The following scheduling criteria re-triggers an Event every five minutes:

```
Schedule EVERY 5 MINUTES
```

Specify Application Parameter Values in a Date-Time/Manual Event

You can define Application parameter values in a Date-Time/Manual Event using an existing Application.

Note: The Application must be defined on the server and have at least one Application parameter.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Date-Time/Manual Event in the Application Events view, and select Edit from the pop-up menu.

   The Event definition dialog opens.

3. Select Application Parameters in the left pane.

   The Application Parameters view opens in the right pane.

4. Click Retrieve from Application to retrieve a list of Application parameters defined in the Application.

   One of the following occurs:

   - If the Parameters table is empty, all of the Application parameters are loaded into the Parameters table.
   - If the Parameters table has parameters, the Application parameters are merged as follows:
     - Only those Application parameters that are defined in the Application but not defined in the Event definition are added to the Parameters table.
     - If the Event definition contains Application parameters that are not defined in the Application, you can choose to remove or retain the Application parameters in the Event definition.

   The Application parameters are added to the Parameters table.

5. Specify values for the retrieved Application parameters.

6. (Optional) Click Add, and enter the following information if you want to add a new Application parameter that is not defined in the Application:

   Parameter Name

   Defines the Application parameter name.

   Value

   Defines the Application parameter value.

7. (Optional) Repeat the previous step to add additional Application parameters.
8. Click OK.

   The Event definition dialog closes.
9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Application parameter values are specified.

More information:

Application Parameters (see page 499)

Remove Application Parameters from a Date-Time/Manual Event

You can remove Application parameters from a Date-Time/Manual Event. You may want to remove Application parameters from an Event for any of the following reasons:

- You have removed or plan to remove the Application parameters from the Application.
- You want to retain only the mandatory Application parameters in the Event.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the Event in the Application Events view, and select Edit from the pop-up menu.

   The Event definition dialog opens.
3. Select Application Parameters in the left pane.

   The Application Parameters view opens in the right pane.
4. Select an Application parameter that you want to remove and click Remove.

   The Application parameter is removed from the Event.

5. (Optional) Repeat the previous step to remove additional Application parameters.
6. Click OK.

   The Event definition dialog closes.
7. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Event is saved.

More information:

Application Parameters (see page 499)

Trigger Workload When a File is Created

You can define a File Trigger Event to trigger workload when a file is created. If the file being monitored is updated after a previous trigger, the Event will be triggered again.

Note: To use File Trigger Events, your system requires CA WA Agent for UNIX, Linux, or Windows.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click within the Events table area in the Application Events view, and select New File Trigger from the pop-up menu.

   The File Trigger Event appears in the list of Events in the Application Events view.
3. Right-click the File Trigger Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.
4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.
5. Click File Trigger in the left pane.

   The File Trigger page opens in the right pane.
6. Complete the following required fields:

   Agent name

   Specifies the name of the agent that monitors the file activity.

File name

Specifies the path to and name of the file to monitor.

Note: If you are connected to the server and the agent is running, you can use the File Browser to browse for the file.

7. (Optional) Specify the following additional information:

Recursive

Monitors for file activity in the specified directory and all of its subdirectories.

When file reaches

Specifies the number of bytes the file size must reach to trigger the Event.

Default: 1 byte

No changes for

Specifies the number of minutes the file must remain unchanged to satisfy the monitor condition.

Default: 0 minutes

Owner user ID

Specifies the UNIX user ID that owns the file to be monitored. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Group

Specifies the UNIX group that owns the file to be monitored.

Monitor as user

Specifies the Windows user ID that monitors the file. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

8. Click OK.

The File Trigger Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

The Event is saved if it contains no errors. The workload is triggered when the file is created.

Example: Trigger Workload When a File is Created

Suppose that a series of files are uploaded to a directory named /uploaded and its subdirectories on the SYSAGENT agent computer. When a file matching the pattern *.tor.eft (for example, test.tor.eft) is uploaded to one of these directories, an Application needs to run to process the file.

To trigger workload when a file is created

1. Enter the following information in the File Trigger page:
   - Agent name—SYSAGENT
   - File name—/uploaded/*.tor.eft
2. Select the Recursive check box to monitor for file activity in the directory and all its subdirectories.
3. Click OK, and save the Event.

Example: Trigger Workload Based on a File Meeting Specified Size and Access Conditions

Suppose that an Application needs to run whenever the /research/analysis file is created on the SYSAGENT agent computer and grows to 100 bytes or higher in size, provided that the file remains unchanged for five or more minutes. If the file is created and, instead, deleted within five minutes, the file trigger should not occur.

If the file is created with an initial size of less than 100 bytes, the file trigger does not occur until the file size reaches 100 bytes and the file remains unchanged for five minutes or more. For example, if the file is created with an initial size of 98 bytes, increases in size to 110 bytes after two minutes, and then remains unchanged, the file trigger occurs after about seven minutes.

To trigger workload based on specified file conditions

1. Enter the following information in the File Trigger page:
   - Agent name—SYSAGENT
   - File name—/research/analysis
   - When file reaches—100
   - No changes for—5
2. Click OK, and save the Event.

More information:

File Trigger Events (see page 500)

Trigger Workload Based on z/OS Data Set Activity

The server can trigger an Event automatically when a data set is created or updated (closed) by a job or a user. You can restrict triggering to data sets created by a specific job, group of jobs, or user ID. The server can also trigger an Event automatically when a file is successfully sent or received by FTP. You can restrict triggering of an Event to a specific host, to a specific user ID, or from a specific logon ID.

Note: To use z/OS Data Set Trigger Events, your system requires CA WA Agent for z/OS.

The server does not trigger a z/OS Data Set Trigger Event when the data set closes during an abnormal termination of a task or job step.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New z/OS Data Set Trigger from the pop-up menu.

   The z/OS Data Set Trigger Event appears in the list of Events in the Application Events view.

3. Right-click the z/OS Data Set Trigger Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click z/OS DataSet Trigger in the left pane.

   The z/OS DataSet Trigger page opens in the right pane.

6. Complete the following required fields:

   Agent name

   Specifies the name of the agent that monitors data set activity.

   Data set name

   Specifies the name of the data set that the agent monitors. You can either specify the name of a single data set or use the hyphen wildcard to specify a group of data sets that match the selection criteria.

   Examples: CYB1.PAYROLL.A, CYB1.PAYROLL.G-

   Note: You cannot use the asterisk (*) as a wildcard.

z/OS User ID

Specifies the z/OS user ID that the monitoring takes place under.

7. (Optional) Specify the following additional information:

Explicit data set trigger

Monitors for an explicit data set notification (used when the data set activity does not generate an SMF record).

FTP

Monitors for a successful FTP transfer.

Host

Specifies a host name to restrict the trigger to FTP transfers to or from a specific remote host.

Logon

Specifies a user ID to restrict the trigger to FTP transfers with a specific logon user ID.

Trigger when action is performed by

Indicates whether to restrict the trigger to data set activity performed by a specific job or user.

Name

Specifies the name of the job or the user who performs the data set activity.

Trigger when dataset is closed if new or updated

Triggers the Event when the data set is closed (created or updated).

Trigger after

Specifies the number of actions that must occur.

8. Click OK.

The z/OS Data Set Trigger Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

The Event is saved if it contains no errors. The workload is triggered when the specified data set activity occurs.

Example: Trigger an Event When a Data Set Is Created or Updated

Suppose that the server must trigger Event PROD.NIGHTLY whenever the data set PROD.CICS.FILE1602 is closed (created or updated). The agent ZOS1 monitors the data set for closure under user CYBDL01.

To trigger an event when a data set is closed

1. Enter the following information in the Properties page:
   - Event prefix—PROD
   - Event name—NIGHTLY
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—PROD.CICS.FILE1602
   - z/OS User ID—CYBDL01
3. Select the Trigger when data set is closed if new or updated check box.
4. Click OK, and save the Event.

Example: Restrict the Trigger to Specific Data Sets Created by a Particular Job

Suppose that the server must trigger Event PROD.PAY_DATA whenever job ABC creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set for creation under user CYBDL01.

To restrict the trigger to specific data sets created by a particular job

1. Enter the following information in the Properties page:
   - Event prefix—PROD
   - Event name—PAY_DATA
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-.
   - z/OS User ID—CYBDL01
3. Select the Job name option button in the Trigger when action is performed by section, and enter **ABC** in the Name field.
4. Click OK, and save the Event.

Example: Restrict the Trigger to Specific Data Sets Created by a Particular User

Suppose that the server must trigger Event PROD.PAY_DATA whenever user CYBER1 creates generation data set USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the data set for creation under user CYBDL01.

To restrict the trigger to specific data sets created by a particular user

1. Enter the following information in the Properties page:
   - Event prefix—PROD
   - Event name—PAY_DATA
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-.
   - z/OS User ID—CYBDL01
3. Select the User ID option button in the Trigger when action is performed by section, and enter **CYBER1** in the Name field.
4. Click OK, and save the Event.

Example: Trigger an Event When a File Is Successfully Received

Suppose that the server must trigger Event PROD.PAY_DATA whenever a file is successfully received from a remote FTP partner, creating a generation of the generation data group USER1.PAYROLL (USER1.PAYROLL.G-). The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To trigger an Event when a file is successfully received

1. Enter the following information in the Properties page:
   - Event prefix—PROD
   - Event name—PAY_DATA
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—USER1.PAYROLL.G-.
   - z/OS User ID—CYBDL01
3. Select Receive from the FTP drop-down list to trigger the Event when a file is successfully received.
4. Click OK, and save the Event.

Example: Trigger an Event When a File is Successfully Sent

Suppose that the server must trigger Event CYBER.XFER whenever data set CYBER.XFER.001 is successfully sent from the local FTP mainframe partner to a remote FTP partner. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To trigger an Event when a file is successfully sent

1. Enter the following information in the Properties page:
   - Event prefix—CYBER
   - Event name—XFER
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS User ID—CYBDL01
3. Select Send from the FTP drop-down list to trigger the Event when a file is successfully sent.
4. Click OK, and save the Event.

Example: Restrict Triggering of an Event to a Specific Host

Suppose that the server must trigger Event CYBER.XFER whenever a remote FTP partner with IP address 172.24.400.200 successfully transfers a file creating the data set CYBER.XFER.001. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering of an Event to a specific host

1. Enter the following information in the Properties page:
   - Event prefix—CYBER
   - Event name—XFER
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS User ID—CYBDL01
   - Host—172.24.400.200
3. Select Receive from the FTP drop-down list to trigger the Event when a file is successfully received.
4. Enter **172.24.400.200** in the Host field.
5. Click OK, and save the Event.

Example: Restrict Triggering of an Event to a Specific User ID

Suppose that the server must trigger Event CYBER.XFER whenever a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming the user ID prefix of the local FTP partner is CYB (CYB-). The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering of an Event to a specific user ID

1. Enter the following information in the Properties page:
   - Event prefix—CYBER
   - Event name—XFER
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS User ID—CYBDL01
3. Select Receive from the FTP drop-down list to trigger the Event when a file is successfully received.
4. Select the User ID option button in the Trigger when action is performed by section, and enter **CYB-** in the Name field.
5. Click OK, and save the Event.

Example: Restrict Triggering of an Event to a Specific Logon ID

Suppose that the server must trigger Event CYBER.XFER whenever a remote FTP partner successfully transfers a file creating the data set CYBER.XFER.001, assuming the remote FTP partner did log on to the FTP server with the CYBER005 user ID. The agent ZOS1 monitors the FTP transfer under user CYBDL01.

To restrict triggering of an Event to a specific logon ID

1. Enter the following information in the Properties page:
   - Event prefix—CYBER
   - Event name—XFER
2. Enter the following information in the z/OS DataSet Trigger page:
   - Agent name—ZOS1
   - Data set name—CYBER.XFER.001
   - z/OS User ID—CYBDL01
3. Select Receive from the FTP drop-down list to trigger the Event when a file is successfully received.
4. Enter **CYBER005** in the Logon field.
5. Click OK, and save the Event.

More information:

z/OS Data Set Trigger Events (see page 501)

Trigger Workload When a JMS Message is Received

You can create a JMS Subscribe Event to trigger workload when a JMS message is received (consumed) from a topic or queue.

Note: To use JMS Subscribe Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Application Services.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New JMS Subscribe from the pop-up menu.

   The JMS Subscribe Event appears in the list of Events in the Application Events view.

3. Right-click the JMS Subscribe Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click JMS Subscribe in the left pane.

   The JMS Subscribe page opens in the right pane.

6. Complete the following required fields:

   Agent name

   Specifies the name of the agent that monitors the topic or queue for JMS messages.

   Initial context 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 the application can use to connect to the application server.

   Example: weblogic.jndi.WLInitialContextFactory

   Provider URL

   Specifies the JNDI service provider URL. For WebLogic servers, the URL is normally in the form t3://*hostaddress*:*port*.

   Example: t3://localhost:7001

Connection 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.

Example: ConnectionFactory

JNDI destination

Specifies the Topic or Queue JNDI name. The job uses the JNDI name to indicate the destination where messages are received.

Example: MyJMSQueue

7. (Optional) Specify the following additional information:

User ID

Specifies the user ID to be used to gain access to the connection factory. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Examples: Bob, Production:Bob

Note: The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.

Job Class

Specifies the job class under which this job runs. 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.

Example: foo

Note: To find out which job classes exist and how many initiators are assigned, ask your agent administrator to check the initiators.class_*n* parameter in the agent parameter file (agentparm.txt).

Filter

Defines the filter used to monitor the topic or queue using regular expression logic.

Example: .*spool.*

Destination type

Specifies whether the job sends to a queue or publishes to a topic.

Example: TOPIC

Output Destination

Specifies the location and file name for the method's output.

8. Click OK.

   The JMS Subscribe Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Event is saved if it contains no errors. The workload is triggered when a JMS message is received (consumed) from a topic or queue.

Example: Trigger Workload When a JMS Message is Received

Suppose that the server must trigger Event MONITOR.MESSAGES whenever a JMS message matching the filter criteria 'abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+' is received from a topic named Topic. The consumed messages are stored in the file /export/home/user1/output1 on the APPSVS agent computer. The Service Provider's URL is t3://172.24.310.660:7001, where 172.24.310.660 is the IP address of the WebLogic Application server and 7001 is the ORB port.

To trigger workload when a JMS message is received

1. Enter the following information in the Properties page:
   - Event prefix—MONITOR
   - Event name—MESSAGES
2. Enter the following information in the JMS Subscribe page:
   - Agent name—APPSVS
   - Initial context factory—weblogic.jndi.WLInitialContextFactory
   - Provider URL—t3://172.24.310.660:7001
   - Connection factory—ConnectionFactory
   - JNDI destination—Topic
   - Filter—'abc\s...\s[a-zA-Z]+\sFilter![\sa-z0-9]+'
   - Destination type—TOPIC
   - Output destination—/export/home/user1/outputfile1
3. Click OK, and save the Event.

More information:

JMS Subscribe Events (see page 502)

Trigger Workload When an SAP Event is Raised

You can create an SAP Event Monitor Event to trigger workload when an SAP event is raised.

Note: To use SAP Event Monitor Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for SAP.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New SAP Event Monitor from the pop-up menu.

   The SAP Event Monitor Event appears in the list of Events in the Application Events view.

3. Right-click the SAP Event Monitor Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click SAP Event Monitor in the left pane.

   The SAP Event Monitor page opens in the right pane.

6. Complete the following required fields:

   Agent name

   Specifies the name of the agent that monitors the raising of an SAP event on the SAP system.

   Name

   Specifies the name of the SAP event to be monitored on the SAP system.

7. (Optional) Specify the following additional information:

   RFC Destination

   Specifies the destination value for the Remote Function Call (RFC) connection and gateway information.

   Note: This field is optional if the agentparm.txt file defines a default destination in the sap.default.destination parameter. If specified, this field overrides the default.

User name

Specifies the user name to use to log on to the SAP system. The user must be defined in the Topology. This field supports the use of a namespace for a user that has more than one password. Contact your administrator for the user name defined in the Topology.

Limits: 32 or fewer characters

Examples: Bob, Production:Bob

Notes:

- The drop-down list displays all the user IDs that are defined in the Topology for the specified agent. You must have at least Read access to the ADMIN.Network Topology permission to view this list.
- This field is optional if the connection properties file defines a default user name. If specified, this field overrides the default.

Client

Specifies a three-digit number that identifies the SAP client.

Example: 800

Note: This field is optional if the connection properties file defines a default client. If specified, this field overrides the default.

Language

Specifies the language to use to log on to the SAP system.

Example: EN for English

Parameter

Specifies the name of the event parameter such as job name or job count.

Note: CA Workload Automation DE and SAP consider event names with different event parameters to be different events.

8. Click OK.

   The SAP Event Monitor Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Event is saved if it contains no errors. The workload is triggered when an SAP event is raised.

Example: Trigger Workload When an SAP Event is Raised

Suppose that the server must trigger Event MONITOR.EVENT whenever the SAP event SAPEVENT with parameter SAPTEST is raised (triggered) on the SAP system. The agent SAPAGENT monitors the raising of the SAP event.

To trigger workload when an SAP event is raised

1. Enter the following information in the Properties page:
   - Event prefix—MONITOR
   - Event name—EVENT
2. Enter the following information in the SAP Event Monitor page:
   - Agent name—SAPAGENT
   - Name—SAPEVENT
   - Parameter—SAPTEST
3. Click OK, and save the Event.

More information:

SAP Event Monitor Events (see page 503)

Trigger Workload When the Number of Rows in a Database Table Changes

You can define a Database Monitor Event to trigger workload when the number of rows that satisfy a monitor condition increases or decreases in a database table.

Note: To use Database Monitor Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

Database Monitor Events monitor for changes using a polling interval (every 10 seconds by default). If you want to detect all changes made to the database, use a Database Trigger Event instead.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New Database Monitor from the pop-up menu.

   The Database Monitor Event appears in the list of Events in the Application Events view.

3. Right-click the Database Monitor Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click Database Monitor in the left pane.

   The Database Monitor page opens in the right pane.

6. Complete the following required fields:

   Agent name

   Specifies the name of the Database Agent that monitors database additions and deletions.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Table name

   Specifies the name of the table to be monitored in the database.

Monitor type

Specifies the type of database change to monitor (INCREASE, DECREASE, INCREASE or DECREASE).

7. (Optional) Specify the following additional information:

DB user

Specifies the database user the Event monitors under. This user overrides the default specified in the agentparm.txt file.

Note: This user and its password must be defined in the Topology under the agent.

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges.

Example: as sysdba

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@host:port:dbname`

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://host:port;DatabaseName=dbname`

For an IBM DB2 database, use the following format:

`jdbc:db2://host:port/dbname`

Monitor condition

Specifies the condition to monitor within the database. This is equivalent to an SQL where clause.

8. Click OK.

The Database Monitor Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

The Event is saved if it contains no errors. The workload is triggered when the specified database table change is detected using the polling interval.

Example: Monitor a Database Table for an Increase in the Number of Rows

Suppose that the DB_AGENT agent monitors the emp table for an increase in the number of rows. When the number of rows with sal greater than 100000 increases, the Event is triggered.

To monitor a database table for an increase in the number of rows

1. Enter the following information in the Database Monitor page:
   - Agent name—DB_AGENT
   - Table name—emp
   - Monitor condition—sal>100000
2. Select INCREASE from the Monitor type drop-down list.
3. Click OK, and save the Event.

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

Suppose that the DB_AGENT agent monitors the STAFF table for a change in the number of rows. When the number of rows that has the name Jonson changes, the Event is triggered. The user entadm is authorized to create triggers on the database.

To monitor a database table for a change in the number of rows

1. Enter the following information in the Database Monitor page:
   - Agent name—DB_AGENT
   - DB user—entadm
   - Table name—STAFF
   - Monitor condition—NAME='Jonson'
2. Select INCREASE or DECREASE from the Monitor type drop-down list.
3. Click OK, and save the Event.

More information:

Database Monitor and Database Trigger Events (see page 504)

Trigger Workload When a Row is Added, Deleted, or Updated in a Database Table

You can define a Database Trigger Event to trigger workload when a row that satisfies a trigger condition is added, deleted, or updated in a database table.

Note: To use Database Trigger Events, your system requires CA WA Agent for UNIX, Linux, or Windows and CA WA Agent for Databases.

A Database Trigger Event creates a database trigger on the database. Contact your database administrator before choosing to use a Database Trigger Event.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New Database Trigger from the pop-up menu.

   The Database Trigger Event appears in the list of Events in the Application Events view.

3. Right-click the Database Trigger Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click Database Trigger in the left pane.

   The Database Trigger page opens in the right pane.

6. Complete the following required fields:

   Agent name

   Specifies the name of the Database Agent that monitors database table changes.

   Note: The drop-down list displays all the agents that are defined in the Topology for the specified job type.

   Table name

   Specifies the name of the table to be monitored in the database.

Trigger type

Specifies the type of change monitored (Insert, Delete, or Update).

Note: You can specify multiple trigger types for Oracle and Microsoft SQL Server. For Oracle, separate multiple types with "or", for example, **Insert or Delete**. For Microsoft SQL Server, separate multiple types with a comma, for example, **Insert, Delete**.

7. (Optional) Specify the following additional information:

DB user

Specifies the database user the Event monitors under. This user overrides the default specified in the agentparm.txt file.

Note: This user and its password must be defined in the Topology under the agent. In addition, this user must be authorized to create triggers on the database. For Microsoft SQL Server, this user must also own the database table that the job monitors for changes.

Oracle DB user type

Specifies the type of Oracle user. In Oracle, a user with sufficient authority can log in with different system privileges.

Example: as sysdba

DB URL

Specifies the database resource location. The server uses Java database connectivity (JDBC) to connect to the database. This URL overrides the default specified in the agentparm.txt file.

For an Oracle database, use the following format:

`jdbc:oracle:thin:@`*`host`*`:`*`port`*`:`*`dbname`*

For an Microsoft SQL Server database, use the following format:

`jdbc:sqlserver://`*`host`*`:`*`port`*`;DatabaseName=`*`dbname`*

For an IBM DB2 database, use the following format:

`jdbc:db2://`*`host`*`:`*`port`*`/`*`dbname`*

Trigger condition

Specifies the condition to monitor within the database. For Oracle and DB2, this is the WHEN clause. For Microsoft SQL Server, this is the IF clause. For the specific database syntax, refer to your database vendor's documentation.

8. Click OK.

   The Database Trigger Event is defined.

9. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

   The Event is saved if it contains no errors. The workload is triggered when the specified database table change occurs.

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

Suppose that the DB_AGENT agent monitors the emp table for an added row or a deleted row. When a row is either added or deleted, the Event is triggered. The Oracle database orcl has a host name of myhost and a port of 1521. The user scott is authorized to create triggers on the database.

To monitor a database table for an added or deleted row

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—scott
   - DB URL—jdbc:oracle:thin:@myhost:1521:orcl
   - Table name—emp
   - Trigger type—Insert or Delete
2. Click OK, and save the Event.

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

Suppose that the DB_AGENT agent monitors the emp table for deleted rows. When a row containing deptno 75 is deleted, an Event is triggered. The Oracle database orcl has a host name of myhost and a port of 1521. The user scott is authorized to create triggers on the database.

To monitor a database table for deleted rows with a trigger condition

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—scott
   - DB URL—jdbc:oracle:thin:@myhost:1521:orcl
   - Table name—emp
   - Trigger condition—old.deptno=75
2. Select Delete from the Trigger type drop-down list.
3. Click OK, and save the Event.

Example: Specify Trigger Conditions for Added Rows of an Oracle Database Table

Suppose that the DB_AGENT agent monitors the emp table for added rows. When a row containing an ename beginning with the letter g is added, the Event is triggered. The Oracle database orcl has a host name of myhost and a port of 1521. The user scott is authorized to create triggers on the database.

To specify trigger conditions for added rows of a database table

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—scott
   - DB URL—jdbc:oracle:thin:@myhost:1521:orcl
   - Table name—emp
   - Trigger condition—new.ename like 'g%'
2. Select Insert from the Trigger type drop-down list.
3. Click OK, and save the Event.

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

Suppose that the DB_AGENT agent monitors the emp table for added or updated rows. When a new or updated row does not contain a job field equal to sales, the Event is triggered. The Oracle database orcl has a host name of myhost and a port of 1521. The user scott is authorized to create triggers on the database.

To monitor a database table for added or updated rows with a trigger condition

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—scott
   - DB URL—jdbc:oracle:thin:@myhost:1521:orcl
   - Table name—emp
   - Trigger type—Insert or Update
   - Trigger condition—new.job<>'sales'

     Note: The <> symbol indicates not equal to.
2. Click OK, and save the Event.

Example: Monitor a Microsoft SQL Table for Added or Deleted Rows with a Trigger Condition

Suppose that the DB_AGENT agent monitors the stores table for an added row or a deleted row. When a row is either added or deleted, the Event is triggered. The Microsoft SQL Server database pubs has a host name of myhost and a port of 1433. The user sa owns the stores table and is authorized to create triggers on the database.

To monitor a database table for added or deleted rows with a trigger condition

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—sa
   - DB URL—jdbc:sqlserver://myhost:1433;DatabaseName=pubs
   - Table name—stores
   - Trigger type—Insert, Delete
2. Click OK, and save the Event.

Example: Monitor a Microsoft SQL Server Table for Two Changes

Suppose that the DB_AGENT agent monitors the sales table for changes to the ord_date and qty columns. When both columns have changed, the Event is triggered. The Microsoft SQL Server database pubs has a host name of myhost and a port of 1433. The user sa owns the sales table and is authorized to create triggers on the database.

To monitor a database table for two changes

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—sa
   - DB URL—jdbc:sqlserver://myhost:1433;DatabaseName=pubs
   - Table name—sales
   - Trigger condition—UPDATE(ord_date) and UPDATE(qty)
2. Select Update from the Trigger type drop-down list.
3. Click OK, and save the Event.

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

Suppose that the DB_AGENT agent monitors the sales table for added rows. When the qty for inserted title_id TC7777 is greater than or equal to 20, the Event is triggered. The Microsoft SQL Server database pubs has a host name of myhost and a port of 1433. The user sa owns the sales table and is authorized to create triggers on the database.

To monitor a database table for added rows with a trigger condition

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—sa
   - DB URL—jdbc:sqlserver://myhost:1433;DatabaseName=pubs
   - Table name—sales
   - Trigger condition—(select QTY from INSERTED where TITLE_ID='TC7777')>=20
2. Select Insert from the Trigger type drop-down list.
3. Click OK, and save the Event.

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

Suppose that the DB_AGENT agent monitors the STAFF table for added rows. When the total number of rows is greater than or equal to 37, the Event is triggered. The DB2 database SAMPLE has a host address of 172.24.4.131 and a port of 50000. The user entadm is authorized to create triggers on the database.

To monitor a database table for added rows with a trigger condition

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—entadm
   - DB URL—jdbc:db2://172.24.4.131:50000/SAMPLE
   - Table name—STAFF
   - Trigger condition—(select count(*) from STAFF)>=37
2. Select Insert from the Trigger type drop-down list.
3. Click OK, and save the Event.

Example: Monitor a DB2 Table for Changes

Suppose that the DB_AGENT agent monitors the STAFF table for changes to the DEPT and JOB columns. When DEPT or JOB is updated, the Event is triggered. The DB2 database SAMPLE has a host address of 172.24.4.131 and a port of 50000. The user entadm is authorized to create triggers on the database.

To monitor a database table for changes

1. Enter the following information in the Database Trigger page:
   - Agent name—DB_AGENT
   - DB user—entadm
   - DB URL—jdbc:db2://172.24.4.131:50000/SAMPLE
   - Table name—STAFF
   - Trigger type—UPDATE of DEPT, JOB
2. Click OK, and save the Event.

More information:

Database Monitor and Database Trigger Events (see page 504)

Trigger Workload When a Global Variable Expression is Met

You can define a Variable Dependency Monitor Event to trigger workload when a global variable expression is met.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click within the Events table area in the Application Events view, and select New Variable Dependency Monitor from the pop-up menu.

   The Variable Dependency Monitor Event appears in the list of Events in the Application Events view.

3. Right-click the Variable Dependency Monitor Event, and select Edit from the pop-up menu.

   The Event definition dialog opens.

4. Enter an Event prefix and an Event name in the Properties page, or leave the defaults as is.

   The Event properties are defined.

5. Click Variable Dependencies in the left pane.

   The Variable Dependencies page opens in the right pane.

6. Click New.

   The New Variable Dependency Expression dialog opens.

7. Complete the following required fields:

   Variable name

   Defines the name of the global variable. This name is not case sensitive. If you are connected to the server, this field is populated with existing variable names.

   Limits: 1-128 alphanumeric or underscore characters. The first character cannot be a number.

   Note: You can specify a variable that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

   Operator

   Specifies the expression operator. Options are the following:

   =

   Specifies the equal operator. The expression evaluates to true if the global variable value equals the specified value.

!=

Specifies the not equal operator. The expression evaluates to true if the global variable is not defined or the global variable value does not equal the specified value.

<

Specifies the less than operator. The expression evaluates to true if the global variable value is less than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 1 < 2, aa < ab, A < a.

>

Specifies the greater than operator. The expression evaluates to true if the global variable value is greater than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 > 1, dd > da, a > A.

<=

Specifies the less than or equal to operator. The expression evaluates to true if the global variable value is less than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 1 <= 2, aa <= ab, A <= a.

>=

Specifies the greater than or equal to operator. The expression evaluates to true if the global variable value is greater than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 >= 1, dd >= da, a >= A.

Contains

Specifies that the expression evaluates to true if the global variable value contains the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to false.

Does Not Contain

Specifies that the expression evaluates to true if the global variable value does not contain the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to true.

Exists

Specifies that the expression evaluates to true if the global variable exists. If the variable does not exist, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

Does Not Exist

Specifies that the expression evaluates to true if the global variable does not exist. If the variable exists, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

8. (Optional) Specify the following additional information:

   Variable Context

   Specifies the name of the context that the global variable belongs to. This name is not case-sensitive. If you are connected to the server, this field is populated with existing context names.

   Limits: 1-128 alphanumeric or underscore characters

   Default: DEFAULT

   Note: You can specify a context that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

   Value

   Defines the value of the global variable.

   Limits: 0-1024 characters

   Default: Empty string

   Examples: final cost, 1000, dept0102, username@company.com

   Note: This field does not apply if the Operator is Exists or Does Not Exist.

9. Click OK.

   The expression appears in the Variable Dependency Expression table on the Variable Dependencies page.

10. (Optional) Repeat steps 6-9 to add additional variable dependency expressions to the Event.

11. Specify the following information:

    Logical operation

    Indicates how multiple variable dependency expressions are evaluated. Options are the following:

    - AND—Indicates that all variable dependency expressions must evaluate to true before the job's variable dependencies are considered met.
    - OR—Indicates that at least one variable dependency expression must evaluate to true before the job's variable dependencies are considered met.

    Default: AND

12. Click OK.

    The Variable Dependency Monitor Event is defined.

13. Right-click the Event in the Application Events view, and select Save from the pop-up menu.

    The Event is saved if it contains no errors. The workload is triggered when the variable expression is met.

Example: Trigger Workload Based on Stock and Market Price

Suppose an online store begins a discount sale whenever the stock reaches 1000 units and the market price falls to $75.

To trigger workload based on stock and market price

1. Enter the following information in the Properties page:
   - Event prefix—MONITOR
   - Event name—DISCOUNT
2. Click New in the Variable Dependencies page, and specify the following information in the New Variable Dependency Expression dialog:
   - Variable context—DEFAULT
   - Variable name—Stock
   - Operator—=
   - Value—1000
3. Click OK.

   The first expression appears in the Variable Dependency Expressions table.
4. Click New in the Variable Dependency page, and specify the following information in the New Variable Dependency Expression dialog:
   - Variable context—DEFAULT
   - Variable name—Price
   - Operator—=
   - Value—75
5. Click OK.

   The second expression appears in the Variable Dependency Expressions table.
6. Select AND from the Logical Operation drop-down list.
7. Click OK, and save the Event.

More information:

Variable Dependency Monitor Events (see page 505)

Chapter 4: Managing Predecessor Dependencies

This section contains the following topics:

Predecessor Dependencies (see page 547)
Edit a Job's Predecessor and Successor Dependencies Using a Dialog (see page 555)
Edit the Release Condition of a Predecessor Dependency (see page 556)
Edit the Release Conditions of a Job's Predecessor and Successor Dependencies (see page 558)
Override Job Inheritance (see page 559)

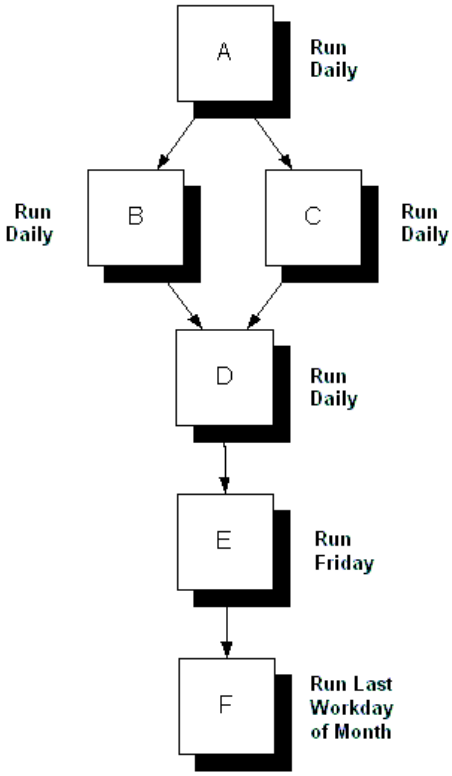
Predecessor Dependencies

Many jobs can run simultaneously in an Application. Predecessor dependencies describe the relationships between jobs in an Application. By default, a job runs after all its predecessors complete successfully. A job that waits for other jobs to complete is in the PREDWAIT state.

Example: Dependencies between jobs in an Application

Every day, jobs A, B, C, and D run. On Fridays, job E runs. On the last workday of the month, job F also runs.

The following illustration displays the run frequency of all the jobs.



The server submits a job after all of the job's predecessors complete successfully. In this Application, the jobs run in the following order:

- Job B and job C run after job A completes successfully
- Job D runs after job B and job C complete successfully
- Job E runs on Fridays after job D completes successfully
- On the last workday of the month, there are two possibilities:
  - If the last workday of the month is a Friday, job F runs after job E completes successfully
  - If the last workday of the month is not a Friday, job F runs after job D completes successfully

More information:

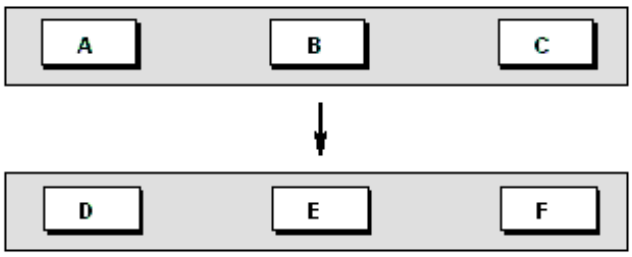
Abandon Predecessor Dependencies at a Specified Time (see page 575)

Interlinked Dependencies

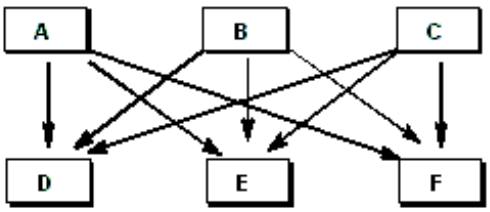
You may have Applications with many groups of interlinked dependencies. You can create dependencies between all the jobs in these groups individually, or you can create a link workload object that waits for the completion of the first group of jobs and then releases the second group of jobs. This approach simplifies the predecessor dependencies and makes it easier to add or delete jobs in each group.

Example: Create a link between Groups of Jobs

Consider the following Application, which has two groups of three jobs each. The second group of jobs (D, E, and F) cannot run until all jobs in the first group (A, B, and C) complete. The following illustration displays the dependency between the two groups of jobs:

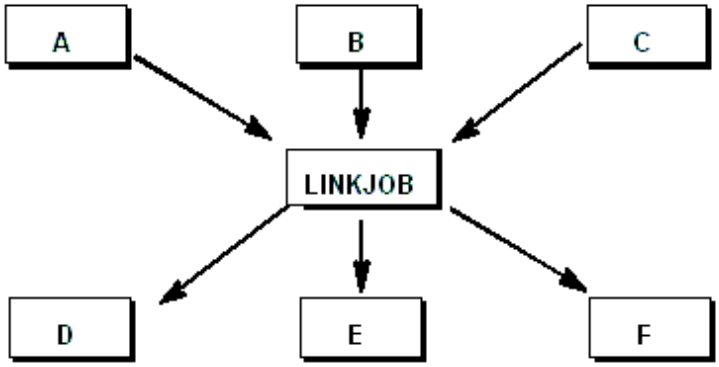


To create this dependency configuration, you must create dependencies from all three jobs in the first group to all three jobs in the second group. The following illustration displays the required dependencies between the two groups:



The interlinked dependencies of this Application may be difficult to read and maintain. To simplify this dependency configuration, create a link that represents the completion of the first group of three jobs and releases the second group of three jobs.

The link waits for jobs A, B, and C to complete and then releases jobs D, E, and F. The following illustration represents the dependencies between the jobs using a link:



The server automatically completes a link as soon as its dependencies are met. After the jobs in the first group (A, B, and C) complete, the server marks the link complete and releases the second group of jobs (D, E, and F).

More information:

Create a Link (see page 201)

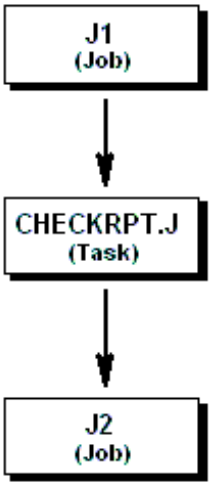
Manual Dependencies

After a job runs, a manual task may need to be performed before the job's successor can run. To represent a manual task, you can set up a task workload object in your Application. A task represents work, such as a manual process that needs to complete before a subsequent job can run. A task does not run on a specific operating system. You must complete a task for its successors to become eligible for submission.

Example: Dependency of a Job on a Manual Task

Suppose that a UNIX job (J1) executes a script to produce a report that you must manually check. To represent the manual checking of the report, you can use a task (CHECKRPT.J). After you check the report, you would mark that task complete, and the server would release the successor job (J2).

The following illustration displays the dependencies between the jobs using a manual task:



More information:

Create a Task (see page 202)

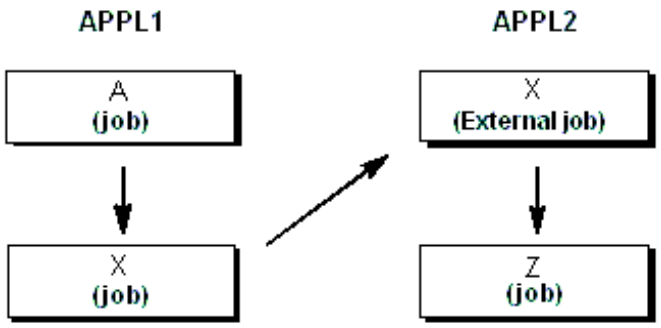
Cross-Application Dependencies

You can set up dependencies between jobs in different Applications using external jobs.

The Application that submits the job is known as the home Application. The Application where the job is defined as external is known as the distant Application. When the job is submitted in the home Application, the server submits the external job in the distant Application. When the job successfully completes in the home Application, the server posts the external job complete in the distant Application and releases its successors.

Example: Create a Cross-Application Dependency

The server submits job X on Fridays as part of Application APPL1. On Fridays, Job Z in Application APPL2 waits for job X. The home Application for job X is APPL1, and the distant Application for job X is APPL2. The following illustration displays the dependency between the two Applications:



When the server generates APPL2 on Fridays, job Z waits until job X completes in its home Application, APPL1.

More information:

Define an External Job (see page 219)

Inherited Dependencies

Jobs in an Application may have different run frequencies. When the server selects jobs for submission, it checks the relationships between jobs and determines whether jobs should inherit dependencies.

Example: Inherited Dependencies of a Job

There are three jobs in an Application. Job A releases job B, which releases job C. Job A and job C run daily, and job B runs only on Friday.

By default, jobs inherit the relationships of their predecessors. Job B runs only on Fridays. On the other days, the server selects job A and job C. Job C inherits the relationship between the unselected job B and job A. On these days, job C depends on job A, not on job B.

You can override job inheritance. Without job inheritance, job A and job C run independently if job B does not run.

The following table summarizes the predecessor dependencies on different days with and without inheritance:

Day	With inheritance	Without inheritance
Days other than Fridays	The server selects job A and job C to run. After job A completes, job C runs.	The server selects job A and job C to run. Job A and job C run independently.
Fridays	The server selects all jobs to run. After job A completes, job B runs. After job B completes, job C runs.	The server selects all jobs to run. After job A completes, job B runs. After job B completes, job C runs.

More information:

Override Job Inheritance (see page 559)

Edit a Job's Predecessor and Successor Dependencies Using a Dialog

With Applications that contain a large number of jobs, it becomes difficult to draw, remove, or view dependencies between jobs that are located a large distance from each other in the workspace. The Edit Job Dependencies feature lets you add, remove, or view a job's predecessor and successor dependencies using a dialog instead.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit Job Dependencies from the pop-up menu.

   The Edit Job Dependencies dialog opens. The Predecessors and Successors list boxes list the job's current predecessors and successors respectively.

3. Edit the job's predecessor or successor dependencies by doing the following.

   - To add predecessor dependencies, select the predecessor jobs you want to add in the Available jobs list box, and click Add under the Predecessors list box.
   - To add successor dependencies, select the successor jobs you want to add in the Available jobs list box, and click Add under the Successors list box.
   - To remove predecessor dependencies, select the predecessor jobs you want to remove in the Predecessors list box, and click Remove under the Predecessors list box.
   - To remove successor dependencies, select the successor jobs you want to remove in the Successors list box, and click Remove under the Successors list box.

   Note: You can press the Ctrl key while making your selections to select multiple jobs at once.

   If new predecessor or successor dependencies are added, the Application is checked for circular dependencies. If the Application contains a circular dependency, an error message appears, and no changes to the job's dependencies are made. Otherwise, the selected jobs are removed from their list box and added to the appropriate list box.

4. Click OK.

   The job's predecessor and successor dependencies are updated as appropriate in the workspace.

5. Click Layout at the bottom of the workspace to organize the layout of your jobs.

   The layout of your jobs is organized.

Edit the Release Condition of a Predecessor Dependency

By default, the server releases a job after its predecessors successfully complete. However, you may want to release a job under a different release condition, such as when a predecessor job fails or when a predecessor completes with a specific exit code.

Note: If you want to view or edit multiple release conditions at once, you can select the release conditions of a job's predecessor and successor dependencies using a single dialog. To use this feature, right-click the job in the workspace, and select Edit Job Dependencies from the pop-up menu.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the dependency line between the jobs for which you want to change the release condition.

3. Select Release Conditions from the pop-up menu.

   The Release Conditions dialog opens.

4. Select one of the following release conditions:

   Failure of the predecessor

   Releases successor if the predecessor fails. If the predecessor completes successfully, the successor is not released.

   Note: If the job goes into the SUBERROR state, the server does not release the job's successors.

   Any completion of the predecessor

   Releases successor whether the predecessor completes successfully or fails.

   Successful completion of the predecessor

   Releases successor if the predecessor completes successfully. If the predecessor fails, the successor is not released. This is the default release condition.

Exit codes

Releases successor based on the predecessor's exit code.

Note: You cannot edit the release condition of a task or a link, since tasks and links can only complete successfully.

5. Click OK.

The job releases its successor based on the release condition you specified.

Note: You can view the release condition of jobs in the Monitor perspective by right-clicking the dependency lines between the jobs and selecting View Release Condition.

Edit the Release Conditions of a Job's Predecessor and Successor Dependencies

By default, the server releases a job after its predecessors successfully complete. However, you can edit the release conditions of a job's predecessor and successor dependencies. For example, you can set the release conditions of a job to release its successors based on specific exit codes, or set the release conditions of a job's predecessors to release the job when the predecessors fail.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit Job Dependencies from the pop-up menu.

   The Edit Job Dependencies dialog opens. The Predecessors and Successors list boxes list the job's current predecessors and successors respectively.

3. Select the dependencies you want to set the release condition for in the Predecessors or Successors list box.

   Note: You can press the Ctrl key while making your selections to select multiple jobs at once.

   If all of the selected dependencies have the same release condition defined, the Release conditions section displays the current release condition. Otherwise, the release condition is not displayed.

4. Select one of the following release conditions from the Release conditions section:

   Failure of the predecessor

   Releases successor if the predecessor fails. If the predecessor completes successfully, the successor is not released.

   Note: If the job goes into the SUBERROR state, the server does not release the job's successors.

   Any completion of the predecessor

   Releases successor whether the predecessor completes successfully or fails.

   Successful completion of the predecessor

   Releases successor if the predecessor completes successfully. If the predecessor fails, the successor is not released. This is the default release condition.

Exit codes

Releases successor based on the predecessor's exit code.

Note: You cannot edit the release condition of a task or a link, since tasks and links can only complete successfully.

5. Click Set.

   The release conditions of all selected predecessor or successor dependencies are set to the specified condition.

6. Click OK.

   The release conditions of the job's predecessor or successor dependencies are updated as appropriate in the workspace.

Override Job Inheritance

You can override job inheritance if you want the jobs in an Application to no longer inherit the relationships of their predecessors.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Do not inherit dependencies check box and click OK.

   The jobs no longer inherit the relationships of their predecessors.

More information:

Inherited Dependencies (see page 553)

Chapter 5: Managing Time Dependencies

This section contains the following topics:

Time Dependencies (see page 562)
Delay Job Submission (see page 569)
Mark a Job Overdue if It Starts Late (see page 571)
Mark a Job Overdue if It Ends Late (see page 572)
Mark a Job Overdue if It Runs Long (see page 573)
Mark a Job Premend if It Ends Prematurely (see page 574)
Abandon Predecessor Dependencies at a Specified Time (see page 575)
Abandon Resource Dependencies at a Specified Time (see page 576)
Abandon Variable Dependencies at a Specified Time (see page 577)
Bypass Jobs at a Specified Time (see page 578)
Propagate Dueout Times (see page 579)

Time Dependencies

A job can contain several time dependencies. When a job reaches its job submission time, the server checks for predecessor, variable, and resource dependencies and submits the job if it meets all the dependencies.

You can use time dependencies to do the following at scheduled times:

- Delay job submission
- Mark a job overdue (Overdue condition)
- Mark a job that ends prematurely (Premend condition)
- Bypass a job
- Abandon a job's predecessor dependencies
- Abandon a job's variable dependencies
- Abandon a job's resource dependencies

You can specify time dependencies by specifying absolute or relative times such as 9PM, 13:00, NOW PLUS 10 MINUTES, or 10PM TODAY PLUS 2 WORKDAYS.

The server resolves your scheduling criteria to a single date and time as follows:

- If you specify a time without a date (for example, 9PM), the server resolves the date to the next occurrence of that time.
- If you specify a date without a time (for example, 1ST MONDAY OF MONTH), the server resolves the time to midnight (00:00:00).
- If you specify a frequency (for example, MONDAYS 10PM), the server resolves the date and time to the next occurrence of that date and time.

Job Submission Time

When an Application is generated, a job's run frequency determines whether the job is selected to run in that generation. To specify when the job runs, you can specify an early submission time when you define the job. The server delays the submission of a job until the submission time you specify. You can delay the job submission to let an Application span many days or weeks.

Note: You cannot specify a time in the run frequency of a job.

If you specify a time (such as 9PM), the server delays submission of the job until the next occurrence of that time. If you do not want to submit the job for the next occurrence of that time, specify a date (such as 9PM JULY 25). The job may still wait for predecessor and resource dependencies, if applicable.

You can also delay the submission of a job for a number of minutes after it is eligible to run. For example, you can specify that the server delay submission of a job for five minutes after it becomes ready for submission.

More information:

Delay Job Submission (see page 569)

Overdue Jobs

If a job does not complete by its anticipated end time, the server adjusts the job's anticipated end time. By default, the server provides no indication that a job has missed its original anticipated end time. If you must monitor when certain jobs are late, specify a dueout time or a maximum execution time. You can specify a dueout time for a late start or late end time. If a job fails to meet its dueout time or exceeds its maximum execution time, the server marks the job with an Overdue condition.

More information:

Anticipated End Times (see page 19)

Dueout Times

The dueout time is the time at which the server marks a job with an Overdue condition because the job missed its anticipated start or end time.

You can specify dueout times for the following conditions:

- If a job has not started by a certain time (late start time)
- If a job has not completed by a certain time (late end time)

For example, you can specify that a job is marked overdue if it does not start by 8 a.m. or does not complete by 1 a.m.

When you set dueout times for a job, you can also provide a notification that the job is late starting or late completing. If a job is late starting or late completing, the server can automatically alert you and others by email or SNMP trap, trigger additional workload, or run a JavaScript script.

More information:

Mark a Job Overdue if It Starts Late (see page 571)
Mark a Job Overdue if It Ends Late (see page 572)

Dueout Times Propagation

The server can propagate dueout times, up-stream, to all predecessors of a job that has a dueout time. The server sets the dueout times of predecessors based on historical average elapsed times, a job's execution time averaged over previous runs. By using dueout propagation, you can avoid specifying dueout times for all jobs in an Application.

The server propagates dueout times when it generates the Application. The following actions affect how the server propagates dueout times in an active Application:

- If you reset a job's late start or late end time, the server resets the late end times of the job's predecessors.
- If you insert a job that is a predecessor to a job with a dueout time, the server resets the late end time of that job based on the inserted job's historical average elapsed time. The server then resets the late end times of the inserted job's predecessors.
- If you request a job that is a predecessor to a job with a dueout time, the server resets the late end time of that job based on the requested job's historical average elapsed time. The server then resets the late end times of the requested job's predecessors.

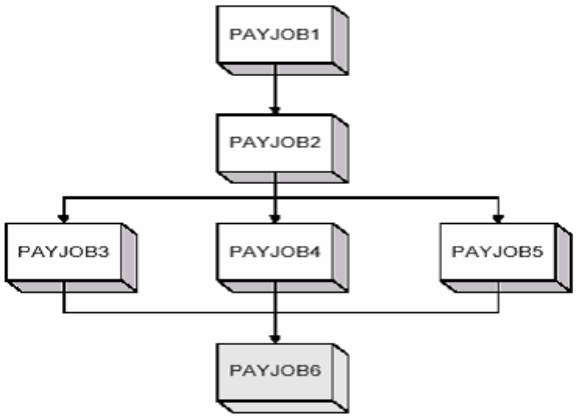
In all cases, the server stops propagating dueout times when it reaches a complete or bypassed predecessor.

Note: The server uses the earliest dueout time when there is a conflict propagating times to a job.

Example: Propagation of Dueout Times

Suppose that PAYJOB6 should start by 7 a.m. PAYJOB6 has an average elapsed time of one hour.

The following illustration displays the relationship between all the jobs:



When the server generates the Application, it sets late end times for all predecessor jobs, PAYJOB1 through PAYJOB5, based on their historical average elapsed times and the late start time of PAYJOB6. If any job in the preceding Application ends late, the server marks the job with an Overdue condition. If you set up a notification in the Application definition for overdue processing, the server sends a notification whenever a job becomes overdue.

In this example, PAYJOB6 has a late start time of 7 a.m., which means PAYJOB3, PAYJOB4, and PAYJOB5 must end by 7 a.m. If PAYJOB3, PAYJOB 4, and PAYJOB5 each have an average elapsed time of one hour, then they will each have a late start time of 6 a.m., and PAYJOB2 will have a late end time of 6 a.m. If PAYJOB2 has an average elapsed time of two hours, then PAYJOB2 will have a late start time of 4 a.m., and PAYJOB1 will have a late end time of 4 a.m. If PAYJOB1 has an average elapsed time of 30 minutes, then PAYJOB1 will have a late end time of 3:30 a.m.

The following table compares the late start times and end times of all jobs if PAYJOB6 has a late start time of 7am or a late end time of 9am:

Job	Average Elapsed Time	If PAYJOB6 has a late start time of 7 a.m.		If PAYJOB6 has a late end time of 9 a.m.	
		Late start time	Late end time	Late start time	Late end time
PAYJOB1	30 minutes	3:30 a.m.	4 a.m.	4:30 a.m.	5 a.m.

Job	Average Elapsed Time	If PAYJOB6 has a late start time of 7 a.m.		If PAYJOB6 has a late end time of 9 a.m.	
		Late start time	Late end time	Late start time	Late end time
PAYJOB1	30 minutes	3:30 a.m.	4 a.m.	4:30 a.m.	5 a.m.
PAYJOB2	2 hrs	4 a.m.	6 a.m.	5 a.m.	7 a.m.
PAYJOB3	1 hr	6 a.m.	7 a.m.	7 a.m.	8 a.m.
PAYJOB4	1 hr	6 a.m.	7 a.m.	7 a.m.	8 a.m.
PAYJOB5	1 hr	6 a.m.	7 a.m.	7 a.m.	8 a.m.
PAYJOB6	1 hr	7 a.m.	8 a.m.	8 a.m.	9 a.m.

More information:

Propagate Dueout Times (see page 579)

Maximum Execution Times

A job's maximum execution time represents a maximum acceptable elapsed execution time in minutes. You can specify a job's maximum execution time as a fixed value or use a calculation to determine the value. If you specify a job's maximum execution time, the server adds this value to the job's start time to obtain an expected latest end time. If the job's actual end time exceeds the expected latest end time, the server marks the job with an Overdue condition.

For example, you can specify that a job that runs on average for one hour is overdue if it completes in more than three hours or if it exceeds its current average execution time by 75%.

When you set the maximum execution time for a job, you can also provide a notification that the job has exceeded its maximum execution time. If a job exceeds its maximum execution time, the server can automatically alert you and others by email or SNMP trap, trigger additional workload, or run a JavaScript script.

More information:

Mark a Job Overdue if It Runs Long (see page 573)

Premend Jobs

You can monitor jobs that end prematurely (Premend condition). By specifying a job's minimum execution time, you can monitor jobs that have ended prematurely and take action.

A job's minimum execution time represents a minimum acceptable elapsed execution time in minutes. You can specify a job's minimum execution time as a fixed value or use a calculation to determine the value. If you specify a job's minimum execution time, the server adds this value to the job's start time to obtain an expected earliest end time. If the job's actual end time is earlier than its expected earliest end time, the server marks the job with a Premend condition.

For example, you can specify that a job that runs on average over two hours ends prematurely if it completes in less than an hour or if it completes in less than 25% of its current average execution time.

When you set the minimum execution time for a job, you can also provide a notification that the job has ended prematurely. If a job ends prematurely, the server can automatically alert you and others by email or SNMP trap, trigger additional workload, or run a JavaScript script.

More information:

Mark a Job Premend if It Ends Prematurely (see page 574)

Delay Job Submission

You can delay job submission by specifying a job submission time. You can also delay the submission of a job for a number of minutes after it is eligible to run.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Complete one of the following fields:

   Do not submit before

   Specifies the job's submission time. The server delays submission of the job until this time.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

   Delay submission when eligible by

   Specifies the number of minutes the server waits to submit a job after it is eligible to run.

   Default: 0 minutes

   Limits: 0-255 minutes

5. Click OK.

   The server delays the job submission as specified.

Example: Delay a Job Until 6 p.m.

Suppose that an Application runs each day at 4 p.m., but a particular job (LATE) within the Application must wait until 6 p.m. An Event is scheduled to run the Application daily at 4 p.m. Job LATE has an early submission time of 6 p.m.

To delay a job until 6 p.m.

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click job LATE in the workspace, and select Edit from the pop-up menu.

   The Basic page of job LATE opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter **6pm** in the Do not submit before field and click OK.

   Job LATE waits until 6 p.m. before running even if job LATE's other dependencies are met before 6 p.m.

Example: Submit a Job at Different Times During the Month

On the last workday of the month, a job runs after 9 p.m. The rest of the month, the job runs after 6 p.m.

The %IF statement for the job's Do not submit before field is as follows:

```
%IF(today('last workday of month'),'9pm','6pm')
```

The today JavaScript built-in function returns true if today is the last workday of the month; otherwise, it returns false.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Example: Delay a Job by 30 Minutes

Job A releases job B in an Application. You want job B to run 30 minutes after job A completes.

To delay a job by 30 minutes after it is eligible to run

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click job B in the workspace, and select Edit from the pop-up menu.

   The Basic page of job B opens.
3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.
4. Enter **30** in the Delay submission when eligible by field and click OK.

   After job A completes, Job B waits for 30 minutes before running.

More information:

Job Submission Time (see page 563)

Mark a Job Overdue if It Starts Late

You can monitor jobs that start too late by specifying a late start time. If the job does not start by the late start time, the server marks the job with an Overdue condition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.
3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.
4. Enter a date and time in the Not started by field.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.
5. Click OK.

   The late start time is specified.

More information:

Dueout Times (see page 564)

Mark a Job Overdue if It Ends Late

You can monitor jobs that complete too late by specifying a late end time. If the job does not complete by the late end time, the server marks the job with an Overdue condition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter a date and time in the Not completed by field.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

5. Click OK.

   The late end time is specified.

More information:

Dueout Times (see page 564)

Mark a Job Overdue if It Runs Long

You can monitor jobs that run too long by specifying a maximum execution time. If the job exceeds its maximum execution time, the server marks the job with an Overdue condition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Specify a maximum execution time in minutes in the Overdue if execution time exceeds field.

5. Click OK.

   The maximum execution time is specified.

Example: Provide Notification if a Job Exceeds its Maximum Execution Time

The server sends an email message if a job exceeds its current average execution time by 75%.

To provide notification if a job exceeds its maximum execution time

1. Define the job with email notification for the Overdue state. Customize the email message if appropriate.

2. Specify the following in the Overdue if execution time exceeds field:

   ```
   %WOB._avgruntime*1.75
   ```

   where the %WOB._avgruntime symbolic variable represents a job's average execution time in minutes based on previous executions.

   If the job exceeds this elapsed time by 75%, the server marks the job with an Overdue condition and sends an email notification to alert others that the job has exceeded its maximum execution time.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

More information:

Maximum Execution Times (see page 567)

Mark a Job Premend if It Ends Prematurely

You can monitor jobs that end prematurely by specifying a minimum execution time. If the job ends before its minimum execution time, the server marks the job with a Premend (premature end) condition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Specify a minimum execution time in minutes in the Premature if execution time is less than field and click OK.

   The minimum execution time is defined.

Example: Provide Notification if a Job Ends Before Its Minimum Execution Time

The server sends an email message if a job completes in less than 25% of its current average execution time.

To provide notification if a job ends before its minimum execution time

1. Define the job with email notification for the Premature end state. Customize the email message, if appropriate.
2. In the Premature if execution time is less than field, specify the following:

   ```
   %WOB._avgruntime*0.25
   ```

   where the %WOB._avgruntime symbolic variable represents a job's average execution time in minutes based on previous executions.

   If the job completes in less than 25% of this elapsed time, the server marks the job with a Premend condition and sends an email notification to alert others that the job has ended before its minimum execution time.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

More information:

Premend Jobs (see page 568)

Abandon Predecessor Dependencies at a Specified Time

When you define a job, you can request the server abandon a job's predecessor dependencies at a specified time.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter a date and time in the Predecessor dependencies field in the Specify time to abandon section.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

5. Click OK.

   The job predecessor dependencies are abandoned at the date and time you specified.

Example: Abandon Predecessor Dependencies at Different Times During the Month

On the last workday of the month, a job must run by 11 p.m., even if its predecessors have not yet completed. The remainder of the month, it must run by 3 a.m., even if its predecessors have not yet completed.

The %IF statement for the job's Predecessor dependencies field in the Specify time to abandon section is as follows:

```
%IF(today('last workday of month'),'11pm','3am')
```

The today JavaScript built-in function returns true if today is the last workday of the month; otherwise, it returns false.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

More information:

Predecessor Dependencies (see page 547)

Abandon Resource Dependencies at a Specified Time

When you define a job, you can specify a time when the server abandons the job's resource dependencies.

Note: If the job has reserved units of a resource, the server makes those resource units available for other jobs.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter a date and time in the Resource dependencies field in the Specify time to abandon section.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

5. Click OK.

   The job resource dependencies are abandoned at the date and time you specified.

More information:

How Jobs with Resource Dependencies are Submitted (see page 581)

Abandon Variable Dependencies at a Specified Time

When you define a job, you can specify a time when the server abandons the job's variable dependencies. At the specified time, the job no longer waits for its variable dependencies to be met. The server then checks that resource dependencies are met and submits the job.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter a date and time in the Variable dependencies field in the Specify time to abandon section.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

5. Click OK.

   The job's variable dependencies are abandoned at the date and time you specified.

More information:

Global Variable Dependencies (see page 591)

Bypass Jobs at a Specified Time

When you define a job, you can request the server bypass the job at a specified time.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

4. Enter a date and time in the Submission field of the Specify time to abandon section.

   Note: To select a time, click the button to the right of the field and select the time from the Time Selection dialog.

5. Click OK.

   The job is bypassed at the date and time you specified.

Example: Bypass a Job at a Certain Time on a Month's First Workday

On the first workday of the month, a job must run before 7:00 p.m. If it has not run by then, it should not run at all. The rest of the month, the job can run when it is ready.

The %IF statement for the job's Submission field in the Specify time to abandon section is as follows:

```
%IF(today('1st workday of month'),'7pm')
```

The today JavaScript built-in function returns true if today is the first workday of the month; otherwise, it returns false.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Propagate Dueout Times

You can propagate dueout times if you want an early warning that jobs in the Application are running late.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Select the Propagate dueout time check box and click OK.

   The server enables anticipated end times and critical path analysis and propagates dueout times in the Application.

Note: If you want this option to apply to all Applications that you define, from the main menu click Window, Preferences, Desktop Client, Define Perspective, Application defaults, and select the Propagate dueout times check box.

More information:

Dueout Times Propagation (see page 565)

Chapter 6: Managing Resource Dependencies

This section contains the following topics:

How Jobs with Resource Dependencies are Submitted (see page 581)
Set Up Resource Dependencies (see page 584)
Set a Job's Priority (see page 586)
Reserve a Resource (see page 586)
Example: Prevent Jobs from Running Concurrently (see page 587)
Example: Control Concurrent Access to a Database (see page 588)
Example: Control When Low-Priority Jobs Run (see page 589)

How Jobs with Resource Dependencies are Submitted

After a job meets all its time and predecessor dependencies, the server does the following:

- Readies the job for submission.
- Checks if the job has requested any resources.
- Checks if resource units are available if the job has requested resources.
  - If all the requested resource units are not available, the job goes into a RESWAIT state.
  - When all the requested resource units are available, the server removes the RESWAIT state and submits the job.

More information:

Abandon Resource Dependencies at a Specified Time (see page 576)

Job Priority

You can prioritize jobs when one or more jobs compete for the same resource. Each job has a priority between 0 and 100 (100 being the highest). The job with the highest value takes priority and obtains the resource first. The default priority is 0. Job priority applies to all active Applications, not just jobs within the same Application.

Note: Priority only applies to renewable and depletable resources.

Example: Job priority assignment

Suppose that independent jobs A, B, and C are waiting for one unit of a renewable resource. If you assign job A a priority of 20, job B a priority of 30, and job C a priority of 40, job C gets the resource first because it has the highest priority, followed by job B, and then job A.

More information:

Set a Job's Priority (see page 586)

Job Reservation

If a job requires more units of a resource than are currently available, the server does not reserve the resource units by default. You can override this default and have the server reserve resources units when the required units are unavailable.

If a job does not reserve resources, a job requiring a lower resource count will receive the available resources first. As a result, jobs that require a large resource count may encounter processing delays, keeping the overall availability resource count low.

If a job reserves resources, the server reserves the resource units that are available at the time the job is next in the queue and holds them while waiting to accumulate the remainder of resource units required for that job.

Note: Only renewable and depletable resources can be reserved.

Example: Resource reservations

Suppose that job A requires 3 units of a resource and job B requires 2 units. Only 2 units of the resource are available and job A precedes job B in the queue.

If job A reserves resources, job A obtains the 2 units of the resource and waits for the third unit. Job B must wait for the 2 units to become available. Job A obtains the resource units before job B, even though job B required fewer resource units.

If job A does not reserve resources, job A must wait for 3 units to become available. Since job B only requires 2 units of the resource and 2 units of the resource are available, job B obtains the resource units first.

More information:

Reserve a Resource (see page 586)

Job Priority and Reservation

A job with a higher priority and the same resource requirements obtains the resource units and runs before a lower priority job. After the higher priority job completes or obtains the resources it needs, the process of gathering resources starts again.

Reserving resources applies only to the request that is first in the queue, which is sorted by priority and request time. For example, if the server receives a resource request for job B before receiving a resource request for job A, it fulfills the resource request for job B first. If two or more jobs requiring the same resource run simultaneously, job resource priorities take precedence.

Set Up Resource Dependencies

You can set up a job's resource dependencies using the Resources page in the job definition to specify the resources the job depends on. A job can request any number of different resources.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Resources in the left pane.

   The Resources page opens in the right pane.

4. Click Add, specify the resource and quantity, and click OK.

   When the job is readied for submission, the server checks if the requested resources are available.

Example: Set Up Three Resource Dependencies for a Job

A job requires two units of a renewable resource named RESREN, one unit of a depletable resource named RESDEP, and four units of a threshold resource named RESTHRESH.

To set up three resource dependencies for a job

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Resources in the left pane.

   The Resources page opens in the right pane.

4. Click Add.

   A new row is added to the Use resources table.

5. Enter RESREN as the resource name and select 2 as the quantity.

6. Click Add.

   A new row is added to the Use resources table.

7. Enter RESDEP as the resource name and select 1 as the quantity.

8. Click Add.

   A new row is added to the Use resources table.

9. Enter RESTHRESH as the resource name and select 4 as the quantity.

10. Click OK.

    When the job is readied for submission, the server checks if the requested resources are available.

Set a Job's Priority

You can set a priority for a job to enable it to obtain resource units before jobs competing for the same resource.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Resources in the left pane.

   The Resources page opens in the right pane.

4. Enter the job's priority between 0 (lowest) and 100 (highest) in the Job priority field and click OK.

   The job priority is set.

More information:

Job Priority (see page 582)

Reserve a Resource

You can reserve resources for a job to enable it to obtain resource units before jobs that require fewer resource units.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Resources in the left pane.

   The Resources page opens in the right pane.

4. Select the Reserve resources check box.

   The job reserves resources.

More information:

Job Reservation (see page 582)

Example: Prevent Jobs from Running Concurrently

You can prevent certain jobs from running at the same time using a renewable resource.

To prevent jobs from running concurrently

1. Define a renewable resource with both availability and maximum availability counts set to 1.
2. Specify that each job requires 1 unit of the renewable resource in the job definition Resources page.

   At any time, either one unit of the resource is available or no units of the resource are available, as follows:

   - If one of the jobs is running, the available count of the resource is zero, and the server prevents the other jobs from running.
   - After the running job completes, the server returns the resource unit to the pool, whether the job was successful or not, and submits one of the other jobs waiting for the resource.

   Note: The jobs you prevent from running concurrently may not be part of the same Application.

Example: Control Concurrent Access to a Database

You can control concurrent access to a database to give specific jobs exclusive use of that database. For example, multiple jobs can read a database at the same time, but only one job can update the database at a time.

To control concurrent access to a database

1. Define a renewable resource with availability and maximum availability counts set to 999.
2. Specify that each read-only job requires 1 unit of the renewable resource in the job definition Resources page.
3. Specify that each update job requires 999 units of the renewable resource in the job definition Resources page.

   The server controls concurrent access to the database, as follows:

   - When an update job runs, no other jobs waiting for the resource can run because an update job requires the maximum count for the resource.
   - Multiple read-only jobs can run concurrently because each read-only job requires only one unit of the resource.

Example: Control When Low-Priority Jobs Run

You can control when low-priority jobs run. For example, a number of low-priority jobs may only run between 8 a.m. and 4 p.m. on workdays. At other times, the server puts the low-priority jobs into a RESWAIT state.

Note: In a multiple CPU environment, you may want to have different time periods to run low-priority jobs on different systems. You can define local resources and set them independently.

To control when low-priority jobs run

1. Define a threshold resource named LOWPRIO with an availability count set to 1.
2. Specify that each low-priority job requires 1 unit of the resource in the job definition Resources page.
3. Do the following to set up an Application that sets the resource count to 1 at the start of the low-priority time, for example, 8am:
   a. Specify a JavaScript script at Event trigger time to set the resource to 1 as follows:

      ```
      resetResourceProperty ('LOWPRIO', 'Availability', '1');
      ```

   b. Schedule an Event to run this Application at the start off the low-priority time.
4. Do the following to set up an Application that sets the resource count to 0 at the end of the low-priority time, for example, 4pm:
   a. Specify a JavaScript script at Event trigger time to set the resource to 0 as follows:

      ```
      resetResourceProperty ('LOWPRIO', 'Availability', '0');
      ```

   b. Schedule an Event to run this Application at the end of the low-priority time.

      Note: The resource name must be in uppercase.

   The server turns the LOWPRIO resource on and off automatically. Any job requiring one unit of the LOWPRIO resource can only run during the low-priority time on workdays.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Chapter 7: Managing Global Variable Dependencies

This section contains the following topics:

Global Variable Dependencies (see page 591)
Set Up a Variable Dependency (see page 592)
Modify a Variable Dependency (see page 597)
Delete a Variable Dependency (see page 600)
%VAR Statement—Specify Global Variables in Job Fields (see page 600)
Example: Specify a Global Variable in a Database Job Definition (see page 602)

Global Variable Dependencies

Jobs can have global variable dependencies. A global variable dependency is a variable expression that must be satisfied before a job is submitted. Such jobs are submitted when all of the global variable dependencies (and the time, predecessor, and resource dependencies) are met, dropped, or abandoned.

For example, you can define a job that only runs when a global variable named quota is assigned a value greater than or equal to 1000. If the global variable dependency is not met at job submission time, the job goes into a VARWAIT state and waits for the dependency to be met, dropped, or abandoned.

There are several ways to manage global variable dependencies. You can set up, edit, and remove variable dependencies by using the Define perspective of CA WA Desktop Client. While monitoring an Application generation in the Monitor perspective, you can set up, edit, remove, and drop variable dependencies and reset the time to abandon variable dependencies. You can also drop variable dependencies by using the CLI. To check the status of jobs in a VARWAIT state, you can evaluate the variable dependencies by using the Monitor perspective or the CLI.

Variable dependencies are stored in the relational database for the server. These dependencies are kept and restored after a warm start of the server or after the server starts as a Primary in a CA WA High Availability configuration. In a cold start of the server, these dependencies are deleted from the relational database.

You can set up global variable dependencies in all job types except for External-Other Scheduler and External-Same Scheduler.

More information:

Abandon Variable Dependencies at a Specified Time (see page 577)

How Jobs with Global Variable Dependencies are Submitted

After a job meets all its time and predecessor dependencies, the server does the following:

- Checks if the job has any global variable dependencies.
  - If the global variable dependencies are not met, the job goes into a VARWAIT state.
  - When all of the global variable dependencies are met, dropped, or abandoned, the server removes the VARWAIT state.
- Checks if the job has any resource dependencies. When the resource dependencies are met, the server submits the job.

Set Up a Variable Dependency

You can set up a job's variable dependencies using the Variable Dependencies page in the job definition. A job can depend on any number of variable dependencies.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.
2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.
3. Click Variable Dependencies in the left pane.

   The Variable Dependencies page opens in the right pane.
4. Click New.

   The Create Variable Dependency Expression dialog opens.
5. Complete the following required fields:

   Variable name

   Defines the name of the global variable. This name is not case sensitive. If you are connected to the server, this field is populated with existing variable names.

   Limits: 1-128 alphanumeric or underscore characters. The first character cannot be a number.

   Note: You can specify a variable that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

Operator

Specifies the expression operator. Options are the following:

=

Specifies the equal operator. The expression evaluates to true if the global variable value equals the specified value.

!=

Specifies the not equal operator. The expression evaluates to true if the global variable is not defined or the global variable value does not equal the specified value.

<

Specifies the less than operator. The expression evaluates to true if the global variable value is less than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 1 < 2, aa < ab, A < a.

>

Specifies the greater than operator. The expression evaluates to true if the global variable value is greater than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 > 1, dd > da, a > A.

<=

Specifies the less than or equal to operator. The expression evaluates to true if the global variable value is less than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 1 <= 2, aa <= ab, A <= a.

>=

Specifies the greater than or equal to operator. The expression evaluates to true if the global variable value is greater than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 >= 1, dd >= da, a >= A.

Contains

Specifies that the expression evaluates to true if the global variable value contains the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to false.

Does Not Contain

Specifies that the expression evaluates to true if the global variable value does not contain the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to true.

Exists

Specifies that the expression evaluates to true if the global variable exists. If the variable does not exist, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

Does Not Exist

Specifies that the expression evaluates to true if the global variable does not exist. If the variable exists, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

Default: =

6. (Optional) Specify the following additional information:

Variable context

Specifies the name of the context that the global variable belongs to. This name is not case-sensitive. If you are connected to the server, this field is populated with existing context names.

Limits: 1-128 alphanumeric or underscore characters

Default: DEFAULT

Note: You can specify a context that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

Value

Defines the value of the global variable.

Limits: 0-1024 characters

Default: Empty string

Examples: final cost, 1000, dept0102, username@company.com

Note: This field does not apply if the Operator is Exists or Does Not Exist.

7. Click OK.

The dialog closes, and the variable dependency expression is added to the Variable Dependency Expressions table.

8. (Optional) Repeat steps 4 to 7 to add additional variable dependency expressions to the job.

9. Specify the following information:

   Logical operation

   Indicates how multiple variable dependency expressions are evaluated. Options are the following:

   - AND—Indicates that all variable dependency expressions must evaluate to true before the job's variable dependencies are considered met.
   - OR—Indicates that at least one variable dependency expression must evaluate to true before the job's variable dependencies are considered met.

   Default: AND

10. Click OK.

    The variable dependency is set up.

Example: Set Up Two Variable Dependencies for a Job

A job requires a global variable named QUOTA to be assigned a value greater than or equal to 1000 and a global variable named SALES to contain the words “final estimate”.

To set up two variable dependencies for a job

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Variable Dependencies in the left pane.

   The Variable Dependencies page opens in the right pane.

4. Click New.

   The Create Variable Dependency Expression dialog opens.

5. Do the following to set up the first variable dependency:

   a. Enter **QUOTA** in the Variable name field.

   b. Select >= from the Operator drop-down list.

   c. Enter **1000** in the Value field.

6. Click OK.

   The dialog closes, and a new row is added to the Variable Dependency Expressions table.

7. Click New.

   The Create Variable Dependency Expression dialog opens.

8. Do the following to set up the second variable dependency:

   a. Enter **SALES** in the Variable name field.

   b. Select Contains from the Operator drop-down list.

   c. Enter **final estimate** in the Value field.

9. Click OK.

   The dialog closes, and a new row is added to the Variable Dependency Expressions table.

10. Leave AND in the Logical operation drop-down list, and click OK.

    When the job is readied for submission, the server checks if the variable dependencies are met.

Modify a Variable Dependency

You can modify an existing variable dependency if you need to change its details.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Variable Dependencies in the left pane.

   The Variable Dependencies page opens in the right pane.

4. Select the variable dependency that you want to modify, and click Edit.

   The Edit Variable Dependency Expression dialog opens.

5. Modify the following required fields as appropriate:

   Variable name

   Defines the name of the global variable. This name must be unique in its variable context. This name is not case sensitive.

   Limits: 1-128 alphanumeric or underscore characters. The first character cannot be a number.

   Note: You can specify a variable that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

   Operator

   Specifies the expression operator. Options are the following:

   =

   Specifies the equal operator. The expression evaluates to true if the global variable value equals the specified value.

   !=

   Specifies the not equal operator. The expression evaluates to true if the global variable is not defined or the global variable value does not equal the specified value.

   <

   Specifies the less than operator. The expression evaluates to true if the global variable value is less than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

   Examples: 1 < 2, aa < ab, A < a.

>

Specifies the greater than operator. The expression evaluates to true if the global variable value is greater than the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 > 1, dd > da, a > A.

<=

Specifies the less than or equal to operator. The expression evaluates to true if the global variable value is less than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 1 <= 2, aa <= ab, A <= a.

>=

Specifies the greater than or equal to operator. The expression evaluates to true if the global variable value is greater than or equal to the specified value. The values are first compared as integers. However, if either of the values contains non-numeric characters, the values are compared lexicographically (alphabetically).

Examples: 2 >= 1, dd >= da, a >= A.

Contains

Specifies that the expression evaluates to true if the global variable value contains the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to false.

Does Not Contain

Specifies that the expression evaluates to true if the global variable value does not contain the specified value. Otherwise, the expression evaluates to false. If the variable is not defined, the expression evaluates to true.

Exists

Specifies that the expression evaluates to true if the global variable exists. If the variable does not exist, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

Does Not Exist

Specifies that the expression evaluates to true if the global variable does not exist. If the variable exists, the expression evaluates to false. A deleted variable does not exist. A variable set to an empty string exists.

6. (Optional) Modify the following additional field as appropriate:

   Variable context

   Specifies the name of the context that the global variable belongs to.

   Limits: 1-128 alphanumeric or underscore characters

   Default: DEFAULT

   Note: You can specify a context that does not exist. This is helpful for setting up a dependency that checks if a specific variable exists.

   Value

   Defines the value of the global variable.

   Limits: 0-1024 characters

   Default: Empty string

   Examples: final cost, 1000, dept0102, username@company.com

   Note: This field does not apply if the Operator is Exists or Does Not Exist.

7. Click OK.

   The dialog closes, and the modified variable dependency expression is displayed in the Variable Dependency Expressions table.

8. (Optional) Specify the following information as appropriate:

   Logical operation

   Indicates how multiple variable dependency expressions are evaluated. Options are the following:

   - AND—Indicates that all variable dependency expressions must evaluate to true before the job's variable dependencies are considered met.
   - OR—Indicates that at least one variable dependency expression must evaluate to true before the job's variable dependencies are considered met.

   Default: AND

9. Click OK.

   The variable dependency is modified.

Delete a Variable Dependency

You can delete a variable dependency if it is no longer in use.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Variable Dependencies in the left pane.

   The Variable Dependencies page opens in the right pane.

4. Select the variable dependency that you want to delete, and click Remove.

   Note: To delete multiple variable dependencies at once, keep the Ctrl key pressed and select each dependency you want to delete, and click Remove.

   The variable dependency is removed from the Variable Dependency Expressions table.

5. Click OK.

   The variable dependency is deleted.

%VAR Statement—Specify Global Variables in Job Fields

You can use global variables when you define and monitor jobs. The %VAR statement lets you specify a global variable name in supported job definition fields in the Define and Monitor perspectives of CA WA Desktop Client. When an Event is triggered, the server tries to substitute the current value of a global variable specified in the %VAR statement. If the variable is not defined at that time, the statement remains unresolved until the job's runtime. At the job's runtime, the server tries to resolve the statement again. If the variable is still undefined, the server submits the job with an unresolved variable.

This statement has the following format:

```
%VAR('name'[,'context'])
```

name

Specifies the name of the global variable. This name is not case sensitive.

Limits: 1-128 alphanumeric or underscore characters. The first character cannot be a number.

context

(Optional) Specifies the name of the context that the global variable belongs to. This name is not case sensitive.

Limits: 0-128 alphanumeric or underscore characters

Default: DEFAULT context

Example: Specify a Global Variable in a Database Job Definition

Suppose that you want to schedule multiple database jobs that use the same IBM DB2 database. Instead of entering the same database resource location in multiple job definitions, you create a global variable named dburl that stores that information. The dburl variable belongs to a global variable context named dbapps.

To specify the global variable in database jobs, enter the following %VAR statement in the DB URL field in the Basic page of the job definition dialog:

```
%VAR('dburl', 'dbapps')
```

When the Event is triggered, the server substitutes the global variable specified in the %VAR statement with the database resource location.

Example: Specify a Global Variable in a Database Job Definition

Suppose that you want to schedule multiple database jobs that use the same IBM DB2 database. Instead of entering the same database resource location in multiple job definitions, you create a global variable named dburl that stores that information. The dburl variable belongs to a global variable context named dbapps.

Note: To use this example, you must have the appropriate security permissions to create global variables in the dbapps context. Without the appropriate permissions, the job goes into a SUBERROR state. To get the appropriate permissions, speak to your CA Workload Automation DE administrator. Your administrator can find more information about the VARIABLE security permission in the *Admin Perspective Help*.

To specify a global variable in a database job definition

1. Define the dburl global variable as follows:
   a. Open the Services perspective.

      A list of server connections is displayed in the Services view.

   b. Right-click Global Variables under your server connection, and click New from the pop-up menu.

      The New Global Variable view opens.

   c. Enter **dburl** in the Variable name field.

   d. Enter **DBAPPS** in the Variable context field.

      Note: The context is converted to upper case as you type it. It is not case sensitive.

   e. Enter the database resource location in the Value field, for example, jdbc:db2://172.24.4.131:50000/SAMPLE.

   f. Click the Save icon.

      The dburl global variable is defined.

2. Specify the global variable in your database jobs using the %VAR statement as follows:
   a. Open the Application that contains your database jobs in the Define perspective.

      The Application appears in the workspace.

   b. Right-click a database job, and select Edit from the pop-up menu.

      The Basic page of the job definition dialog opens.

   c. Enter **%VAR('dburl', 'dbapps')** in the DB URL field.

   d. Click OK.

      The job is updated.

e. Repeat the previous steps for each of your database jobs.

f. Upload the Application to the server.

3. Simulate the Application to review the resolution of the global variable as follows:

   a. Right-click the Event you want to simulate in the Event Triggers workspace, and select Simulate from the pop-up menu.

      The Simulate Event dialog opens.

   b. Complete the fields as appropriate and click OK.

      The simulation opens in a separate window. On the right pane, the details of the job definitions appear. In the database job definitions, the value of DB URL resolves to the value of the dburl global variable, for example, jdbc:db2://172.24.4.131:50000/SAMPLE.

      Note: If the job details do not appear, from the main menu click Window, Preferences, Desktop Client, Services Perspective, Simulate, and select the View Job Details text check box.

   When the Event is triggered, the server substitutes the global variable specified in the %VAR statement with the database resource location.

Chapter 8: Working with Notifications

This section contains the following topics:

Notifications (see page 605)
Set Up an Email Notification in the Job Definition (see page 610)
Set Up an Email Notification in the Application (see page 613)
How to Set Up an Alert Notification (see page 616)
Example: Resubmit a Job if it Completes with a Particular Exit Code Before a Specific Time (see page 626)
Set Up an SNMP Notification in the Job Definition (see page 628)
Set Up an SNMP Notification in the Application (see page 630)

Notifications

Notifications let you automatically monitor a job's progress and notify users or take action when the job reaches a certain state or condition. You can set up notifications for different job-processing monitor states, such as when a job completes, fails, or becomes overdue. Notifications let you automate your workload processing.

You can set up the following three types of notifications:

Email Notifications

Sends an email when a job reaches any of the states specified in the notification. For example, you can set up a notification to email an operator when a particular job fails or to provide status for a long-running Application.

Alert Notifications

Triggers an Event or runs a JavaScript script when the job reaches any of the states specified in the notification. For example, you can set up an Alert to trigger an Event to start an Application when the last job on another Application completes or run a JavaScript script to complete an Application when a job is overdue.

SNMP Notifications

Sends a Simple Network Management Protocol (SNMP) trap to Network Management Stations (NMSs) when a job reaches any of the states specified in the notification. For example, you can send an SNMP trap to an SNMP-enabled agent or an SNMP Manager when a particular job fails.

Note: For more information about SNMP traps, see the *Implementation Guide*.

To send a notification when a job reaches a certain state, you should set up the notification in the job definition. If you want the notification to apply to all jobs in the Application, you should instead set up the notification in the Application. Notifications in the job definition override notifications in the Application for the same monitor state.

Example: Override Email Notification at the Application Level

Suppose that an Application contains an email notification that sends an email to operator1 for job failure and submission errors. Job A in this Application contains an email notification that sends an email to operator2 when it fails. The following occurs if job A fails or has a submission error:

- If job A fails, the server sends the email notification to operator2, as referenced in the job definition.
- If job A has a submission error, the server sends an email to operator1, as referenced in the Application.

Monitor States

You can set up notifications to notify users or act when a job reaches a certain state. You can set up notifications for the following job states and conditions:

Abandon submission

Sends a notification when the submission of a job is abandoned.

Note: This state requires that you specify an abandon time for job submission.

Agent down

Sends a notification when the agent referenced in the job definition is down.

Complete

Sends a notification when the job completes successfully.

Exec

Sends a notification when the job starts to execute.

Failed

Sends a notification when the job fails according to its exit code settings.

Force completed

Sends a notification when the job is forced complete.

Global variable wait

Sends a notification if the job is waiting for one or more variable dependencies to be met.

Inactive

Sends a notification if the job is inactive.

Note: This state can only be used with Oracle Applications Single Request jobs.

Monitor

Sends a notification when the condition that the job is monitoring has occurred.

Note: This state can be used with jobs that support continuous monitoring such as File Trigger, Database Monitor, and Database Trigger.

Overdue

Sends a notification when the job is overdue.

Note: This condition requires that you specify an overdue time for job start or completion, or a maximum execution time.

Premature end

Sends a notification when the job ends prematurely.

Note: This condition requires that you specify a minimum execution time.

Ready

Sends a notification when all predecessor dependencies and submit times are met and the job is not held.

Note: The server does not monitor for resource dependencies.

Resource wait

Sends a notification if the job is waiting for one or more resource dependencies to be met.

Suberror

Sends a notification when a submission error occurs, such as when a script is not found.

Unknown

Sends a notification when there is a configuration or system error that the server does not recognize.

Note: Depending on the job type, only a subset of these job states may be available.

Email Notifications

When a job reaches a certain state, the server can send an email notification to notify users about the job's progress. For example, you can set up a notification to email an operator when a particular job fails or to provide status for a long-running Application.

You can add as many email notifications to a job definition or Application as you need. When the job reaches the state you have specified, the server sends an email notification to each address provided. Every email notification includes a message that contains the following details:

- Job name
- Application name
- State
- Exit code, if assigned
- Agent name, if applicable
- The name of the file being run, if applicable
- (Optional) A customized message
- (Optional) The job's spool file

You can send email notifications to any device that accepts email messages, including pagers and cell phones.

More information:

Set Up an Email Notification in the Job Definition (see page 610)
Set Up an Email Notification in the Application (see page 613)

Alert Notifications

An Alert notification can trigger additional workload automatically. When a job reaches the state specified in the Alert notification, the server automatically triggers the Event defined in the Alert. For example, when the last job of an Application completes, an Alert can trigger an Event to run another Application. You can trigger an Event for all jobs or specific jobs in an Application.

An Alert notification can also run a JavaScript script automatically. When a job reaches the state specified in the Alert notification, the server automatically runs the JavaScript script defined in the Alert.

An Alert can run a JavaScript script to execute commands such as the following:

- Resubmitting a job
- Completing a job or Application
- Dropping job dependencies
- Dropping resource dependencies
- Resetting time dependencies
- Bypassing a job
- Holding and releasing jobs or Applications
- Resetting resource availability counts

You can store the scripts for these actions in a central repository.

For example, you can set up an Alert to run a JavaScript script to complete an Application when a job is overdue or resubmit a job if it completes with a particular exit code before a specific time.

In the JavaScript script that the Alert runs, you can use built-in variables. When the Alert runs, the server sets those variables. These variables are only available in the JavaScript script that specific Alert runs. You can assign these variables to other variables so that other JavaScript scripts can use them.

SNMP Notifications

SNMP notifications help agents in a network inform Network Management Stations (NMSs) that something unexpected happened, such as a job failure, or communicate status information, such as an agent start. The CA Workload Automation DE server sends SNMP traps automatically when the server or an agent starts or stops.

Using SNMP notifications, you can send SNMP traps to an SNMP-enabled agent or an SNMP manager when a job reaches a specific monitor state.

More information:

Set Up an SNMP Notification in the Job Definition (see page 628)
Set Up an SNMP Notification in the Application (see page 630)

Set Up an Email Notification in the Job Definition

If you want an email notification to send an email when a job reaches a certain condition, such as Fail or Overdue, set up the email notification in the job definition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the Email tab.

5. Clear the Use Application-level defaults check box.

   If the Application contains any email notifications, the notifications are copied to the job definition. You can edit or remove a copied notification by selecting the notification and clicking Edit or Remove as required.

   Notes:

   - If you leave the Use Application-level defaults check box selected, the email notifications defined in the Application are displayed, but you cannot edit or add new notifications.
   - To suppress the Application-level notifications for this job, ensure that the list of notifications is empty.

6. Click New to add a notification.

   The New Email Notification dialog opens.

7. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

Monitor states

Specifies the job states you can monitor.

To

Specifies the recipient's email address.

Note: After you specify an email address, you must click Add to add the email address to your notification. You can add multiple email addresses.

Subject

(Optional) Specifies a customized email subject.

Notes:

- To insert the default subject, select the Use default subject check box. The administrator can edit the default subject by modifying the default email template in the Admin perspective.
- To specify a customized email subject, clear the Use default subject check box. You can insert Application- and job-level built-in symbolic variables into the subject line by right-clicking in the field and selecting the symbolic variable you want to insert from the drop-down menu.

Message

(Optional) Adds a customized message to the email.

Notes:

- To insert the default message, select the Use default message option button. The administrator can edit the default message by modifying the default email template in the Admin perspective.
- To add to the default message, select the Attach to default message option button and enter your message. To specify a customized message or override the default message, select the Override default message option button and enter your message. You can insert Application- and job-level built-in symbolic variables into the message by right-clicking in the text box and selecting the symbolic variable you want to insert.

Attach spool file

(Optional) Attaches the job's spool file to the email notification.

Notes:

- The administrator can configure a shared parameter to limit the size of spool file attachments in email when providing email notifications for failed jobs. For more information about server shared parameters, see the *Admin Perspective Help*.
- Not all job types support the Attach spool file option.

8. Click OK.

   The New Email Notification dialog closes.

9. Click OK.

   The server sends an email notification whenever the job reaches the monitor state you specified.

More information:

Monitor States (see page 606)
Email Notifications (see page 608)

Set Up an Email Notification in the Application

If you want the email notification to apply to all jobs in the Application, set up the email notification in the Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the Email tab and click New.

   The New Email Notification dialog opens.

5. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

   Monitor states

   Specifies the job states you can monitor.

   To

   Specifies the recipient's email address.

   Note: After you specify an email address, you must click Add to add the email address to your notification. You can add multiple email addresses.

   Subject

   (Optional) Specifies a customized email subject.

   Notes:

   - To insert the default subject, select the Use default subject check box. The administrator can edit the default subject by modifying the default email template in the Admin perspective.
   - To specify a customized email subject, clear the Use default subject check box. You can insert Application- and job-level built-in symbolic variables into the subject line by right-clicking in the field and selecting the symbolic variable you want to insert from the drop-down menu.

Message

(Optional) Adds a customized message to the email.

Notes:

- To insert the default message, select the Use default message option button. The administrator can edit the default message by modifying the default email template in the Admin perspective.
- To add to the default message, select the Attach to default message option button and enter your message. To specify a customized message or override the default message, select the Override default message option button and enter your message. You can insert Application- and job-level built-in symbolic variables into the message by right-clicking in the text box and selecting the symbolic variable you want to insert.

Attach spool file

(Optional) Attaches the job's spool file to the email notification.

Notes:

- The administrator can configure a shared parameter to limit the size of spool file attachments in email when providing email notifications for failed jobs. For more information about server shared parameters, see the *Admin Perspective Help*.
- Not all job types support the Attach spool file option.

6. Click OK.

   The New Email Notification dialog closes.

7. Click OK.

   The server sends an email notification whenever a job in the Application reaches the monitor state you specified.

Example: Send an Email Notification When a Job Fails or Has an Error

Suppose that you want to send an email notification to operator1@yourcompany.com and operator2@yourcompany.com when a job in the Application fails or has a monitoring or submission error.

To send an email notification when a job fails or has an error

1. Open the New Email Notification dialog.
2. Select the Failed and Suberror monitor states.
3. Enter **operator1@yourcompany.com,operator2@yourcompany.com** in the To field and click Add.
4. Clear the Use default subject check box and enter **Job has failed** in the Subject field.
5. Select the Use default message option button.
6. Select the Attach spool file check box and click OK twice.

   The server sends an email notification if a job in the Application fails or has a monitoring or submission error.

More information:

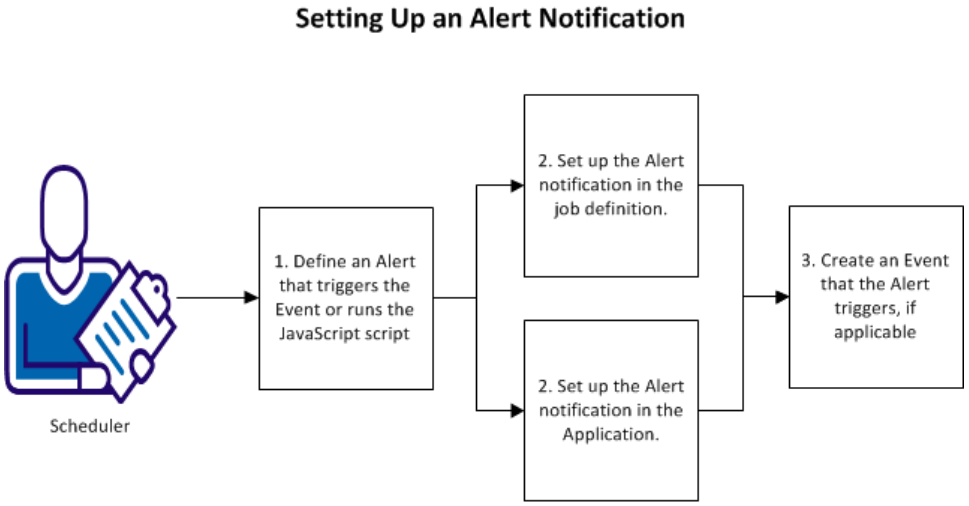
Monitor States (see page 606)
Email Notifications (see page 608)

How to Set Up an Alert Notification

As a scheduler, you can set up notifications to monitor a job's progress and notify users or act when the job reaches a monitor state. You can set up notifications for different job-processing monitor states, such as when a job completes, fails, or becomes overdue. Notifications let you automate your workload processing.

An Alert notification can trigger more workload or run a JavaScript script automatically. When a job reaches the state specified in the notification, the server triggers the Event or runs the JavaScript script defined in the Alert.

This scenario walks you through the process of setting up an Alert notification:



To set up an Alert notification, follow these steps:

1. Do one of the following tasks:
   - Define an Alert that triggers the Event (see page 617).
   - Define an Alert that runs the JavaScript script (see page 618).
2. Set up the Alert notification in the job definition (see page 620) or Application (see page 623).
3. Create an Event that the Alert triggers, if applicable (see page 624).

Example: Set Up an Alert Notification

Some examples of when you would set up an Alert notification include the following:

- Trigger an Event that runs a second Application when the last job of a specific Application completes
- Complete an Application if a job does not start by a specific time
- Resubmit a job if a job completes with a particular exit code before a specific time

More information:

Alert Notifications (see page 609)

Define an Alert that Triggers the Event

You can define an Alert to trigger an Event when a job reaches a particular state.

Follow these steps:

1. Connect to the server as a scheduler using CA WA Desktop Client.
2. Open the Services perspective.

   A list of server connections is displayed in the Services view.
3. Right-click Alerts under your server connection, and click New from the pop-up menu.

   The New Alert view opens.
4. Enter the Alert name in the Alert name field.
5. Select the Trigger an Event option button.
6. Enter the Event prefix and Event name, and click the Save icon.

   The Alert is defined.

Example: Start an Application When Another Application Completes

In this example, you will trigger an Event that runs a second Application when the last job of the first Application completes. First, define an Alert to trigger the Event.

1. Connect to the server as a scheduler using CA WA Desktop Client.
2. Right-click Alerts in the Services perspective, and select New from the pop-up menu.

   The New Alert dialog opens.
3. Enter **ALERTA** in the Alert name field.
4. Select the Trigger an Event option button.
5. Enter **MYEVENT** in the Prefix field, enter **EVENTB** in the Name field, and click Save.

   The Alert is defined. The next step is to set up an Alert notification in the last job of the first Application.

More information:

Create an Event that the Alert Triggers (see page 624)

Define an Alert that Runs the JavaScript Script

You can define an Alert to run a JavaScript script when a job reaches a particular state.

Follow these steps:

1. Connect to the server as a scheduler using CA WA Desktop Client.
2. Open the Services perspective.

   A list of server connections is displayed in the Services view.
3. Right-click Alerts under your server connection, and click New from the pop-up menu.

   The New Alert view opens.
4. Enter the Alert name in the Alert name field.
5. Do *one* of the following actions:

   - To run a script from the repository, select the Run a JavaScript from Repository option button, and select the script from the Name drop-down list.

     The script appears in the editor, but it cannot be edited. Skip to the last step.
   - To import a script from your local computer or a network drive, select the Run this JavaScript option button, click Import, browse to the script, and click Open.

     The imported script appears in the editor.
   - To define a script, select the Run this JavaScript option button, and type the script content in the editor.
6. (Optional) Press Ctrl+spacebar in the editor to invoke content assist.

   A pop-up displays the built-in JavaScript variables and functions that you can use in the script. The pop-up also displays the native JavaScript functions and objects.

   Notes:

   - You can filter the variables, objects, and functions that appear in the pop-up. For example, if you type **file** and invoke content assist, the pop-up displays all server built-in functions that begin with file. You can further filter the list in the pop-up by typing additional text, for example, **_l**.
   - To display the description of a built-in JavaScript variable or function, select the variable or function in the pop-up. You cannot display the description of a native JavaScript function or object. For more information about the native JavaScript functions and objects, visit the ECMA International website (www.ecma-international.org) and search for the ECMA-262 Standard for scripting languages.
   - The pop-up displays all global variables and functions. To display all variables and functions that belong to an object or context, type the object or context name followed by a period in the editor. For example, if you type **APPL.**, a pop-up displays all the APPL built-in variables and functions.

7. Select a variable or function in the list and press Enter to insert it in the script.

   The selected variable or function appears in the editor.

8. (Optional) Click Check Syntax to validate the syntax of the script.

   Errors and warnings, if any, are displayed in a dialog. The errors are also highlighted in the editor.

   Note: After you correct the script, click Check Syntax again. Repeat until you do not receive any errors or warning messages. Click OK to close the dialog.

9. (Optional) Click Export to save the script on your local computer or a network drive. Browse to the location where you want to save the script, enter a file name, and click Save.

10. Click the Save icon to save the Alert.

    The Alert is defined.

Note: For more information about JavaScript expressions and built-in functions, see the *Programming Guide*.

Example: Complete an Application if a Job is Overdue

In this example, you will run a JavaScript script that completes an Application if a job is overdue. First, define an Alert to run the JavaScript script.

1. Open the Services perspective.

   A list of server connections is displayed in the Services view.

2. Right-click Alerts under your server connection, and click New from the pop-up menu.

   The New Alert dialog opens.

3. Enter **ALERTA** in the Alert name field.

4. Select the Run this JavaScript option button.

5. Enter the following script:

   ```
   execCommand('ALL','APPLC.0','ACTION COMPLETE');
   ```

6. Click the Save icon.

   The Alert is defined. The next step is to set up an Alert notification and overdue time dependency in the job definition.

More information:

JavaScripts (see page 633)

Set Up an Alert Notification in the Job Definition

If you want the Alert notification to send an Alert when a job reaches a certain condition, such as Fail or Overdue, set up the Alert notification in the job definition. Notifications in the job definition override notifications in the Application for the same monitor state.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the Alerts tab.

5. Clear the Use Application-level defaults check box.

   If the Application contains any Alert notifications, the notifications are copied to the job definition. You can edit or remove a copied notification by selecting the notification and clicking Edit or Remove as required.

   Notes:

   - If you leave the Use Application-level defaults check box selected, the alert notifications defined in the Application are displayed, but you cannot edit or add new notifications.
   - To suppress the Application-level notifications for this job, ensure that the list of notifications is empty.

6. Click New to add a notification.

   The New Alert Notification dialog opens.

7. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

   Monitor states

   Specifies the job states you can monitor.

Alert Name

Specifies the name of the Alert.

8. Click OK.

   The New Alert Notification dialog closes.

9. Click OK.

   The server sends an Alert notification whenever the job reaches the monitor state you specified.

Example: Start an Application When Another Application Completes

After you define the Alert, set up the Alert notification in the last job (LASTJOB) of the Application.

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Double-click the job LASTJOB, and click Notifications in the left pane.

   The Notifications page opens in the right pane.

3. Clear the Use Application-level defaults check box.

4. Select the Alerts tab and click New.

   The New Alert Notification dialog opens.

5. Select the Complete check box in the Monitor states section.

6. Select the Alert ALERTA from the Alert name drop-down list, and click OK twice.

   The Alert notification is set up. The final step is to create an Event that the Alert triggers.

Example: Complete an Application if a Job is Overdue

After you define the Alert, set up the Alert notification and overdue time dependency in the job definition.

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Double-click the job, and click Notifications in the left pane.

   The Notifications page opens in the right pane.

3. Clear the Use Application-level defaults check box.

4. Select the Alerts tab and click New.

   The New Alert Notification dialog opens.

5. Select Overdue in the Monitor states section.

6. Select the Alert ALERTA from the Alert name drop-down list and click OK.

7. Click Time Dependencies in the left pane.

   The Time Dependencies page opens in the right pane.

8. Enter **10pm** in the Not started by field.

9. Click OK.

   If the job does not start by 10 p.m., an Alert triggers. The Alert runs a JavaScript script that uses the execCommand built-in function to complete the current generation (generation 0) of Application APPLC.

More information:

Alert Notifications (see page 609)
Monitor States (see page 606)

Set Up an Alert Notification in the Application

If you want the Alert to apply to all jobs in the Application, set up the Alert in the Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the Alerts tab and click New.

   The New Alert Notification dialog opens.

5. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

   Monitor states

   Specifies the job states you can monitor.

   Alert Name

   Specifies the name of the Alert.

6. Click OK.

   The New Alert Notification dialog closes.

7. Click OK.

   The server sends an Alert notification whenever the job in the Application reaches the monitor state you specified.

Example: Set Up an Alert When a Job Completes

Suppose that the server triggers an Alert named ALERT1 whenever a job in the Application completes.

To set up an Alert when a job completes

1. Open the New Alert Notification dialog.
2. Enter ALERT 1 as Alert name, select Complete in the Monitor state field, and click OK twice.

   The server triggers an Alert whenever a job in the Application completes.

More information:

Alert Notifications (see page 609)
Monitor States (see page 606)

Create an Event that the Alert Triggers

If you want the Alert to trigger an Event when the job reaches the states specified in the Alert notification, you must define the Event.

Follow these steps:

1. Open the Services perspective.

   A list of server connections is displayed in the Services view.
2. Right-click Events under your server connection, and click Open from the pop-up menu.

   The Events view opens.
3. Right-click in the Event listing, and select New DateTime/Manual from the pop-up menu.

   The New DateTime/Manual view opens.
4. Enter the Event prefix and Event name referenced in the Alert.
5. Specify the Application to run in the Application field, and click the Save icon.

   Note: Since the Alert triggers the Event automatically, you do not need to enter scheduling criteria for the Event.

   The Event that the Alert triggers is created.

Example: Start an Application When Another Application Completes

After setting up the Alert notification in the job definition, create the Event that the Alert triggers.

1. Create a DateTime/Manual Event in the Services perspective.
2. Enter **MYEVENT** as the Event Prefix, **EVENTB** as the Event Name, and **APPLB** as the Application name.
3. Click Save.

   When LASTJOB in Application APPLA completes, it triggers Alert ALERTA. ALERTA triggers the Event MYEVENT.EVENTB to run the Application APPLB.

More information:

Define an Alert that Triggers the Event (see page 617)

You have successfully set up an Alert notification. When a job reaches the state specified in the notification, the server triggers the Event or runs the JavaScript script defined in the Alert.

Example: Resubmit a Job if it Completes with a Particular Exit Code Before a Specific Time

Suppose that the server monitors a job (JOB1) in an Application (APPLD). If the job completes before 10 p.m. with an exit code of 4 (considered a job failure), you want the server to resubmit the job.

To resubmit a job if it completes with a particular exit code before a specific time

1. Define an Alert to run the JavaScript script as follows:
   a. Open the Services perspective.

      A list of server connections is displayed in the Services view.

   b. Right-click Alerts under your server connection, and click New from the pop-up menu.

      The New Alert dialog opens.

   c. Enter **ALERTC** in the Alert name field.

   d. Select the Run this JavaScript option button.

   e. Enter the following script:

```
if (WOB._LTIME < '22.00.00')
  execCommand ('%WOB._name','%(APPL._name).%APPL._gen',
  'ACTION RESUB');
```

   f. Click the Save icon.

2. Set up an Alert notification in the job definition as follows:
   a. Open the Application in the Define perspective.

      The Application appears in the workspace.

   b. Double-click the job, and click Notifications in the left pane.

      The Notifications page opens in the right pane.

   c. Select the Alerts tab.

   d. Clear the Use Application-level defaults check box and click New.

      The New Alert Notification dialog opens.

   e. Enter **4** in the Return code field, select Complete in the Monitor states section, and select ALERTC from the Alert name drop-down list.

   f. Click OK twice.

   If the job completes with an exit code of 4, an Alert triggers to run a JavaScript script. The script uses the WOB.LTIME symbolic variable to check when the Alert was triggered. If the Alert was triggered before 10 p.m., the script uses the execCommand built-in function to resubmit the job.

   Note: Built-in variables are used to ensure the job is resubmitted in the correct generation of the correct Application. This lets you reuse the same technique for any number of jobs regardless of the Applications to which they belong.

Set Up an SNMP Notification in the Job Definition

If you want the SNMP notification to send a SNMP notification when a job reaches a certain condition, such as Fail or Overdue, set up the SNMP notification in the job definition.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the SNMP Traps tab.

5. Clear the Use Application-level defaults check box.

   If the Application contains any SNMP notifications, the notifications are copied to the job definition. You can edit or remove a copied notification by selecting the notification and clicking Edit or Remove as required.

   Notes:

   - If you leave the Use Application-level defaults check box selected, the SNMP notifications defined in the Application are displayed, but you cannot edit or add new notifications.
   - To suppress the Application-level notifications for this job, ensure that the list of notifications is empty.

6. Click New to add a notification.

   The New SNMP Trap Notification dialog opens.

7. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

   Monitor states

   Specifies the job states you can monitor.

SNMP enabled agent

(Optional) Specifies the name of an SNMP-enabled agent to sent the SNMP trap to.

Note: If you leave this field blank, the server broadcasts the SNMP trap to the SNMP manager configured for the server.

SNMP message

Specifies the SNMP message.

Note: You can insert Application- and job-level built-in symbolic variables into the SNMP message by right-clicking in the field and selecting the symbolic variable you want to insert.

8. Click OK.

   The New SNMP Trap Notification dialog closes.

9. Click OK.

   The server sends a SNMP message whenever the job reaches the monitor state you specified.

More information:

Monitor States (see page 606)
SNMP Notifications (see page 609)

Set Up an SNMP Notification in the Application

If you want the SNMP notification to apply to all jobs in the Application, set up the SNMP notification in the Application.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click Notifications in the left pane.

   The Notifications page opens in the right pane.

4. Select the SNMP Traps tab and click New.

   The New SNMP Trap Notification dialog opens.

5. Complete the following fields as appropriate:

   Return code

   (Optional) Specifies an exit code to send the notification on a more specific condition.

   Note: This field applies to only Complete and Fail monitor states.

   Monitor states

   Specifies the job states you can monitor.

   SNMP enabled agent

   (Optional) Specifies the name of an SNMP-enabled agent to sent the SNMP trap to.

   Note: If you leave this field blank, the server broadcasts the SNMP trap to the SNMP manager configured for the server.

SNMP message

Specifies the SNMP message.

Note: You can insert Application- and job-level built-in symbolic variables into the SNMP message by right-clicking in the field and selecting the symbolic variable you want to insert.

6. Click OK.

   The New SNMP Trap Notification dialog closes.

7. Click OK.

   The server sends an SNMP message whenever the job reaches the monitor state you specified.

Example: Send an SNMP Trap When the Job Submission is Abandoned

Suppose that you want to set up an SNMP notification so that the server sends an SNMP trap whenever the submission of a job in the Application is abandoned.

To set up an SNMP notification

1. Open the New SNMP Trap Notification dialog.
2. Select Abandon submission as the Monitor state.
3. Enter the following message in the SNMP message field:

   ```
   Submission of job %WOB._name in Application %APPL._name has been abandoned in generation %APPL._gen.
   ```

4. Click OK twice.

   The server sends the SNMP notification if a job in the Application is abandoned.

More information:

Monitor States (see page 606)
SNMP Notifications (see page 609)

Chapter 9: Working with JavaScripts

This section contains the following topics:

JavaScripts (see page 633)
Store a JavaScript Script in the Application (see page 634)
Specify JavaScript Scripts in the Event Definition (see page 636)
Specify a JavaScript Script in the Application Definition (see page 637)
Specify a JavaScript Script in the Job Definition (see page 638)
%IF Statement—Create Conditional Schedule Criteria (see page 640)

JavaScripts

You can use a JavaScript script within an Application or Alert to perform many types of operations.

Some common uses for a script include the following:

- Using conditional logic to specify a variation in the schedule
- Defining symbolic variables
- Using server built-in functions

You can use a script to perform an operation within the Application definition or within one or more jobs. Before the script can be available to the Application or job, you must first store it in the Application that will use the script or in the JavaScript repository.

You can also use a script to perform an operation within the Alert definition. You can define the script in the Alert definition or store it in the JavaScript repository.

The name of each script must be unique for every instance of the server.

Store a JavaScript Script in the Application

If the JavaScript script is used in only one Application, you can store the script in the Application. If you want to use the script in multiple Applications, store the script in the JavaScript repository instead.

Follow these steps:

1. Open the Application in the Define perspective.
2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.
3. Click JavaScript in the left pane.

   The JavaScript page opens in the right pane.
4. Click Edit Local JavaScript.

   The Application Defined (Local) JavaScript Scripts dialog opens.
5. Click Add to name the script.

   The Adding new JavaScript dialog opens.
6. Define a meaningful name for the script in the JavaScript name field and click OK.

   Note: The script name must be unique.

   The script name appears in the left pane.
7. Type the script content in the editor in the right pane.

   Note: You can import a script from your computer or a network drive by clicking Import.
8. (Optional) Press Ctrl+spacebar in the editor to invoke content assist.

   A pop-up displays the server built-in JavaScript variables and functions that you can use in the script. The pop-up also displays the native JavaScript functions and objects.

   Notes:

   - You can filter the variables, objects, and functions that appear in the pop-up. For example, if you type **file** and invoke content assist, the pop-up displays all server built-in functions that begin with file. You can further filter the list in the pop-up by typing additional text, for example, **_l**.
   - To display the description of a built-in JavaScript variable or function, select the variable or function in the pop-up. You cannot display the description of a native JavaScript function or object. For more information about the native JavaScript functions and objects, visit the ECMA International website (www.ecma-international.org) and search for the ECMA-262 Standard for scripting languages.

- The pop-up displays all global variables and functions. To display all variables and functions that belong to an object or context, type the object or context name followed by a period in the editor. For example, if you type **APPL.**, a pop-up displays all the APPL built-in variables and functions.

9. Select a variable or function in the list and press Enter to insert it in the script.

   The selected variable or function appears in the editor.

10. (Optional) Click Check Syntax to validate the syntax of the script.

    Errors and warnings, if any, are displayed in a dialog. The errors are also highlighted in the editor.

    Note: After you correct the script, click Check Syntax again. Repeat until you do not receive any errors or warning messages. Click OK to close the dialog.

11. (Optional) Click Export to save the script on your local computer or a network drive. Browse to the location where you want to save the script, enter a file name, and click Save.

12. Click OK.

    The script is stored in the Application.

Specify JavaScript Scripts in the Event Definition

You can specify a list of JavaScript scripts in the Event definition, which are run when the Event is triggered. You can specify scripts in an Event definition to set default values for the Application, define symbolic variables to be used by the Application, and for many other purposes.

Note: You cannot define JavaScript scripts in the Event definition. You can only run scripts defined in the JavaScript repository.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Double-click the Event in the Event Triggers workspace.

   The Properties page of the Event trigger definition dialog opens.

3. Click JavaScript in the left pane.

   The JavaScript page opens in the right pane.

4. Click Add.

   A new row is added to the Specify JavaScript references table.

5. Click the drop-down arrow in the new row.

   A list of scripts that are defined in the JavaScript repository is displayed in a pop-up list.

   Note: You can also type the script name in the row. Before the Event triggers, the script must be defined in the JavaScript repository and you must have access to the script.

6. Select a script from the list.

   The script appears in the area to the right of the table.

7. (Optional) Repeat the above steps to specify additional scripts within the Event definition.

8. Click OK.

   The specified scripts run when the Event is triggered.

Note: Specifying a script in the Event definition is equivalent to specifying the script in the Application definition using the At Event trigger time option.

Specify a JavaScript Script in the Application Definition

You can specify a JavaScript script in the Application definition, and run it when the Event is triggered or when each job in the Application starts to run. You can specify a script in an Application definition to set default values for the Application, define symbolic variables to be used by the Application, and for many other purposes.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the Application in the Application Workspace view, and select Properties from the pop-up menu.

   The Basic page of the Application properties dialog opens.

3. Click JavaScript in the left pane.

   The JavaScript page opens in the right pane.

4. Specify the script to run in the Run JavaScript section:

   At Event trigger time

   Runs the script when the Event triggers. Use this option if the script uses symbolic variables that are resolved at Event trigger time such as Application name, default Agent name, job names and qualifiers, run frequencies, notifications, resource dependencies, and time dependencies.

   Note: If you want to run multiple scripts at Event trigger time, specify the scripts in the Event definition instead.

   At run time

   Runs the script each time a job runs within the Application. Use this option if the script uses symbolic variables that are resolved at run time such as Agent name, command or script name, arguments, user ID, exit codes, and environment variables.

   Note: For more information about choosing when to run the script, see the *Programming Guide*.

   To view the selected script, click View beside the script name.

   To edit the script defined in the Application, click Edit Local JavaScript.

   To view the scripts in the JavaScript repository, click Show JavaScript Repository.

5. Click OK.

   The script runs when the Event is triggered (Event trigger time) or when the jobs in the Application run (run time).

Specify a JavaScript Script in the Job Definition

You can specify a JavaScript script in a job definition, and run it when the Event is triggered or when the job runs. You can specify a script in a job definition to create symbolic variables for use by the job, specify alternative times to submit the job, and for other purposes.

Follow these steps:

1. Open the Application in the Define perspective.

   The Application appears in the workspace.

2. Right-click the job in the workspace, and select Edit from the pop-up menu.

   The Basic page of the job definition dialog opens.

3. Click JavaScripts in the left pane.

   The JavaScripts page opens in the right pane.

4. Ensure that the At Event trigger time and the At job run time sections are expanded.

5. Specify the script to run in the JavaScript section:

   At Event trigger time

   Runs the script when the Event triggers. Use this option if the script uses symbolic variables that are resolved at Event trigger time such as Application name, default Agent name, job names and qualifiers, run frequencies, notifications, resource dependencies, and time dependencies.

At job run time

Runs the script each time a job runs within the Application. Use this option if the script uses symbolic variables that are resolved at run time such as Agent name, command or script name, arguments, user ID, exit codes, and environment variables.

Note: This option does not apply to external jobs.

Note: For more information about choosing when to run the script, see the *Programming Guide*.

6. Specify a script in one or both of these sections in one of the following ways:

   - Select a script from the Script name drop-down list. This list displays the JavaScript scripts defined in the Application and in the global JavaScript repository. To view a selected script, click View beside the Script name drop-down list.
   - Type in the script in the Define or Import JavaScript to run text box. No syntax validation or syntax highlighting is done on this script. This script will be local to the job.
   - Click Import and browse for a predefined script on your computer. This script will be local to the job.

   Note: You can define up to two scripts in a job definition provided the scripts run at different times. For example, you can define a local script to run at Event trigger time and select a script from the JavaScript repository to run at job run time.

7. Click OK.

   The script runs when the Event is triggered (Event trigger- time) or when the job runs (job run time).

%IF Statement—Create Conditional Schedule Criteria

Use the %IF statement to add conditional logic to your Application and job definitions without having to run a script. For example, you can use the %IF statement to submit a job at a different time on the last workday of the month. When the server generates an Application (trigger time), it evaluates the %IF statement for a true or false condition and returns the appropriate value.

Note: You cannot use the %IF statement when inserting a job or resetting a job definition or time dependency.

This statement has the following format:

```
%IF(logical_test,'value_if_true'[,'value_if_false'])
```

logical_test

Defines a JavaScript logical expression or JavaScript function that returns a true or false (Boolean) value.

Note: The server evaluates the logical expression at run time. You cannot use runtime variables (APPL._R* or WOB._R*) in the logical expression. Use a JavaScript script instead and invoke it at run time.

value_if_true

Defines any character string or JavaScript variable that the server returns if the logical test is true.

Note: If the true value contains special characters (such as backslashes or single quotes), precede them with a backslash (\).

value_if_false

(Optional) Defines any character string or JavaScript variable that the server returns if the logical test is false.

Note: If the true value contains special characters (such as backslashes or single quotes), precede them with a backslash (\).

Example: Run a Job on the Last Workday of the Month Except in December

A job runs on the last workday of the month, unless the month is December. If it is December, the job runs the day before the last workday of the month.

The %IF statement for the job's run frequency When field is as follows:

```
%IF(APPL._SMM == '12','last workday of month less 1 day','last workday of month')
```

The APPL._SMM symbolic variable represents the number of the scheduled month.

Example: Run a Job on Even Days

A job runs only on even-numbered days of the month. The conditional logic uses the day number to determine whether a day is an even-numbered day or not. The % (remainder) operator divides the day by 2 and tests the remainder. If the remainder is 0, the server sets the run frequency value to day.

The %IF statement for the job's run frequency When field is as follows:

```
%IF(APPL._SDD%2 == 0,'today')
```

The APPL.SDD symbolic variable represents the number of the scheduled day of the month.

Example: Run a Job on Certain Weeks of a Month

A job runs Monday through Saturday during the third and fourth weeks of October.

The %IF statement for the job's run frequency When field is as follows:

```
%IF(APPL._SMM == '10','anyday of 3rd 4th week of month except sunday')
```

The APPL._SMM symbolic variable represents the number of the scheduled month.

Note: For more information about creating conditional schedules, using runtime variables, and running scripts at run time, see the *Programming Guide*.

Chapter 10: Working with Schedule Criteria

This section contains the following topics:

Schedule Criteria (see page 644)
Days of the Week and Months (see page 645)
Dates (see page 646)
Times (see page 650)
Time Zones (see page 651)
Daylight Saving Time (see page 653)

Schedule Criteria

CA Workload Automation DE has a built-in understanding of general scheduling terms. You can add your own unique scheduling terms, special processing periods, holidays, and other special days using calendars. You use free-format, everyday English as scheduling terms to specify schedule criteria. You can test your criteria and review the results.

You can use schedule criteria to perform the following actions:

- Schedule, trigger, and simulate Events
- Schedule the suspending and resuming of Events
- Indicate a job's run frequency
- Set and reset a job's time dependencies
- Create forecast reports
- Trigger Events using the execTrigger JavaScript function
- Generate time and date variables using the genTime JavaScript function

The scheduling terms are derived from plain English terminology. The syntax is not case sensitive so, for example, CA Workload Automation DE accepts Sat, sat and sAt. You may also use words and phrases other than specific dates and times to define schedule criteria, such as first workday of month, 2nd Friday of year, and so on.

The following lists some common scheduling terms:

- daily
- workdays
- holidays
- 9:00am
- Nov 6, 2006
- monday wednesday friday
- daily except Mondays
- Last workday of month
- last day of month less 2 days
- every 2 weeks starting Monday
- 6am daily
- hourly starting 2pm today
- 9am workdays
- 15th day of March June August

Days of the Week and Months

CA Workload Automation DE recognizes the seven days of the week.

You can specify the days of the week in schedule criteria in the following ways:

- Use the full name, such as WEDNESDAY.
- Shorten the name to three characters, such as SUN, THU, or FRI.
- Use the plural form of the name, such as MONDAYS and FRIDAYS.
- Enter multiple days of the week, such as MONDAY WEDNESDAY FRIDAY.
- Refer to every day of the week as DAILY or ANYDAY.
- Refer to Monday through Friday as WEEKDAYS.
- Refer to workdays (as defined by your calendar) excluding holidays as WORKDAYS.

CA Workload Automation DE recognizes the twelve months of a year. You can abbreviate any month name to three letters, such as APR instead of APRIL.

Examples: Schedule Criteria for Various Days of the Week and Month Combinations

The following table displays examples of schedule criteria categorized by occurrence:

Occurrence	Schedule Criteria to Enter
Monday, Wednesday, and Friday	Mon Wed Fri
Everyday except Mondays	daily except Mondays
Every other workday beginning this Friday	every 2 workdays starting Friday
Last workday of each week	Last workday of week
Every Saturday in June	Saturday of June
Every Monday, unless Monday is a holiday. If Monday is a holiday, run on the previous workday.	monday less 0 workdays
Every Monday, unless Monday is a holiday. If Monday is a holiday, run on the next workday.	monday plus 0 workdays
Every Monday, unless Monday is a holiday.	monday except holidays
2nd last workday of each month	last workday of month less 1 workday
3rd last day of each month	last day of month less 2 days
Fifth last workday of each month	last workday of month less 4 workdays

Occurrence	Schedule Criteria to Enter
First Friday of each month. If this is not a workday, then run on the next workday after the first Friday of the month	first Friday of month plus 0 workdays
Every other Monday beginning next Monday	every 2 weeks starting Monday plus one week
Every other Monday beginning in a week from next Monday	every 2 weeks starting Monday plus 1 week
Last complete weekend (Saturday and Sunday) of each month	anyday of last complete weekend of month
Monday and Tuesday in June, July, and August	Monday Tuesday of June July August or Monday Tuesday June July August

Dates

You can express the date in various formats as follows:

- Ordinal numbers
- Days of month
- Range of days
- Implied period
- Date format

Ordinal Numbers

An ordinal defines an item's position in a series. The ordinal qualifiers are as follows:

- st, for example 1st
- nd, for example 2nd
- rd, for example 3rd
- th, for example 4th

You can use ordinal numbers with scheduling terms, for example 2ND MONDAY OF YEAR. You can use more than one ordinal, as in 3RD AND 4TH MONTH OF YEAR.

For example, specify the first Monday of each month as 1ST MONDAY OF MONTH or 1ST MONDAY MONTHLY. 10AM 3RD MONTHLY requests activity at 10 a.m. on the third day of each month.

If you use an ordinal number with a scheduling term (other than with the starting parameter) without stating what the schedule term refers to, CA Workload Automation DE assumes the following defaults:

Schedule Term	Default
date, day name, or workday	of month
week or month	of year
holiday	of month

For example, if you specify 3RD FRIDAY AT 9AM, the schedule occurs at 9 a.m. on the third Friday of every month, while 9AM LAST HOLIDAY occurs on the last holiday of the calendar month at 9 a.m.

Days of Months

CA Workload Automation DE recognizes a number joined to the name of a month as a day of the month, for example JUN3, 22OCT, 6NOV2006.

You can also use the ordinal qualifiers st, nd, rd, or th with a month name, as in JANUARY21ST or AUGUST29th. Alternatively, you can specify JAN 21 or 29th AUGUST.

Range of Days

You can use a phrase that describes a range of inclusive dates. To separate the date range, you must use a hyphen (-). For example, you can specify 3RD-6TH DAY OF MONTH.

Implied Periods

The following calendar terms are implied periods:

- week
- weekend
- month
- year

For example, ANYDAY OF 2ND WEEKEND OF THE MONTH refers to the second weekend of all the months, 3RD DAY OF LAST WEEK OF YEAR refers to the third day of last week of any year, and 5TH WORKDAY OF 2ND WEEK OF YEAR refers to the fifth workday of the second week of any year.

Date Format

CA Workload Automation DE recognizes the date format *yyyy*/*mm*/*dd* or *yyyy-mm-dd*, where yyyy is the year number, mm is the month number, and dd is the day number.

The following table shows how you would express January 2, 2006 using these two formats:

Format	Example
yyyy/mm/dd	2006/01/02 or 2006/1/02
yyyy-mm-dd	2006-01-02 or 2006-1-02

Examples: Schedule Criteria for Various Date Combinations

The following table displays examples of schedule criteria categorized by occurrence:

Occurrence	Schedule Criteria to Enter
4 p.m. each day between November 6, 2006 and April 22, 2007	4pm daily starting Nov 6,2006 ending 4:01pm Apr 22,2007 **Note:** To ensure the ending date is included, include a time on the ending date.
Every Saturday in June starting in the year 2006	Saturday of June starting June 1 2006
8 a.m. on the last Saturday of each month, beginning in October 2000	8:00 last saturday of month starting October 1, 2000

Occurrence	Schedule Criteria to Enter
15th day of March, June, and August	15th day of March June August
First weekday (Monday through Friday) on or after the 15th day of the month	15th day of month plus 0 weekdays
Last day of month only in months that have 31 days	31st day within month
February 29th in leap years	29th day within Feb **Note:** Without the "within" keyword, Events scheduled to run on February 29 run on March 1 in non-leap years.
Friday after the 1st Sunday of the month	1st Sunday of month plus 5 days
First Friday after the 3rd Thursday of each month	3rd Thursday of this month plus 1 day **Note:** If the month starts on a Friday, this criterion will not be the 3rd Friday of the month.
First Saturday of the month. If this is the first day of the month, run on second Saturday of month instead.	Saturday 2nd monthly **Note:** CA Workload Automation DE interprets the preceding expression as the first Saturday on or after the 2nd day of the month.
10 a.m. on the 3rd, 13th, and 23rd day of month	10am 3rd 13th 23rd day of month
6 a.m. daily from the 10th to 20th day of each month inclusive	6am 10th - 20th day of month
3rd, 9th, 10th, 11th, 12th, 14th workday of each month	3rd 9th - 12th 14th workday of month
Third and fourth Sundays of month if four Sundays in month; otherwise, third and fifth Sundays of month if five Sundays in month	3rd and last Sunday of month
Each year on July 23	July 23 yearly
300th day of 2002	2002.300
December 21, 2002	2002/12/21

Times

CA Workload Automation DE recognizes any time of the day. CA Workload Automation DE accepts *hh*:*mm*:*ss*:*SSS* time-of-day formats , where *hh* = hours, *mm* = minutes, *ss* = seconds, and *SSS* = milliseconds.

The following table displays an example of the various time formats:

Format	Example
hh.mm	12.30
hh:mm	12:30
hh:mm.ss	12:30.15
hh:mm:ss	12:30:15
hh:mm.ss.tt	12:30.15.10

The following rules apply to times in schedule criteria:

- Any number followed by am or pm implies a time, such as 6PM, 7AM, 9:00AM.
- NOON (12:00) and MIDNIGHT (00.00) are recognized as times of the day.
- MIDNIGHT TUESDAY or 24:00 TUESDAY are equivalent to 00:00 WEDNESDAY.
- 24:00 WEEKDAYS and MIDNIGHT WEEKDAYS are equivalent to 00:00 TUE, WED, THU, FRI, SAT.
- 12AM is equivalent to 00:00, while 12PM is equivalent to 12:00 or NOON.

Note: You cannot specify a time in the run frequency of a job.

Examples: Schedule Criteria for Various Time Combinations

The following table displays examples of schedule criteria categorized by occurrence:

Occurrence	Schedule Criteria to Enter
6 a.m. each day	6am daily
Every 30 minutes beginning at 2 p.m. today	every 30 minutes starting 2pm today
every hour starting today at 2 p.m.	hourly starting 2pm today
9 a.m. on workdays	9am workdays
8 a.m. on the 3rd workday of each month	8am 3rd workday of month

Time Zones

To schedule workload based on different time zones, you identify the time zone based on its abbreviation or a major world city in that time zone.

Note: The specified time zone is only used to evaluate the schedule criteria before it is converted to the server's time zone. The schedule follows the daylight saving changes of the server's time zone, *not* the specified time zone. If the specified time zone and the server's time zone have a different daylight saving period, the scheduled times displayed in CA WA Desktop Client (converted to the local computer's time zone) will vary according to the rules of the server's time zone.

The following table displays the time zones you can identify based on their local time abbreviations:

Abbreviation	Long Form
ATZ	Atlantic Time Zone
CTZ	Central Time Zone
ETZ	Eastern Time Zone
GMT	Greenwich Mean Time
MTZ	Mountain Time Zone
PTZ	Pacific Time Zone
UTC	Universal Coordinated Time

The following table displays the time zones you can identify based on major world cities:

Time Zone Code	City
amsterdam	Amsterdam, Netherlands
anchorage	Anchorage, United States
athens	Athens, Greece
bangkok	Bangkok, Thailand
berlin	Berlin, Germany
brisbane	Brisbane, Australia
brussels	Brussels, Belgium
budapest	Budapest, Hungary
cairo	Cairo, Egypt

Time Zone Code	City
dhaka	Dhaka, Bangladesh
hong_kong	Hong Kong, China
honolulu	Honolulu, United States
jerusalem	Jerusalem, Israel
kabul	Kabul, Afghanistan
karachi	Karachi, Pakistan
kolkata	Kolkata, India
kuala_lumpur	Kuala Lumpur, Malaysia
kuwait	Kuwait City, Kuwait
london	London, England
moscow	Moscow, Russia
muscat	Muscat, Oman
nairobi	Nairobi, Kenya
paris	Paris, France
perth	Perth, Australia
prague	Prague, Czech Republic
rome	Rome, Italy
seoul	Seoul, South Korea
shanghai	Shanghai, China
singapore	Singapore
sydney	Sydney, Australia
taipei	Taipei, Taiwan
tehran	Tehran, Iran
tokyo	Tokyo, Japan

Examples: Time Zone Specification

The following are examples of schedule criteria for time zones:

- 8PM CTZ refers to 8 p.m. Central Time Zone
- 9AM ROME refers to 9 a.m. in Italy
- GMT + 2:00 refers to 2 hours after Greenwich Mean Time

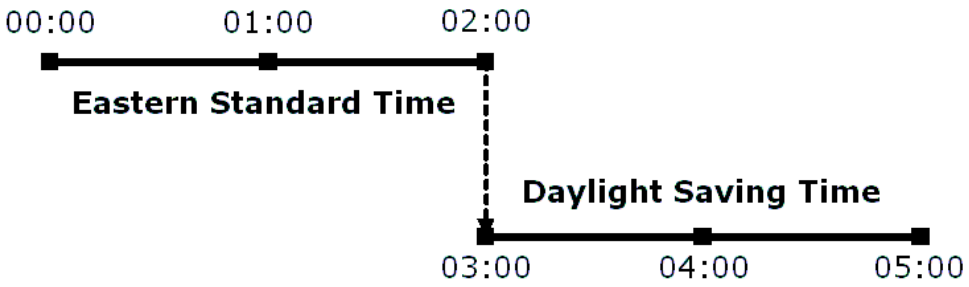
Daylight Saving Time

CA Workload Automation DE observes daylight saving time (DST) depending on where it is installed in the world.

For example, in Sydney, Australia, DST is observed; spring forward occurs at 2:00 a.m. on the last Sunday of October and fall back occurs at 3:00 a.m. on the last Sunday of March. Some places, such as Phoenix, USA and Regina, Canada, do not observe DST.

Spring-Forward Adjustments

In the North American Eastern Standard Time Zone, DST starts on the second Sunday of March at 2:00 a.m.; the time springs forward one hour to 3:00 a.m. The following illustration displays DST spring forward:



Events that are scheduled to run during the spring-forward transition period (2:00 a.m. to 3:00 a.m. on the day spring forward occurs) run immediately after the transition at 3:00 a.m. For example, Event A is scheduled at 2:30 a.m. on March 11, 2007. At 01:59:59 a.m., the time springs forward to 3:00 a.m. At 3:00 a.m., the server considers Event A overdue and triggers the Event immediately.

At the job level, if a job has a submission time of 2:30 a.m., the server submits the job at 3:00 a.m. (assuming all other dependencies have been met). For example, Event B is scheduled daily, suspended at 2:30 a.m., and resumed at 5:00 a.m. On March 11, 2007, the server suspends the Event at 3:00 a.m. and resumes the Event at 5:00 a.m.

The server adjusts the times for the following to 3:00 a.m. if they fall in the spring-forward transition period:

- Event simulation for the current year
- Event trigger
- Event schedule criteria (resume and suspend times)

  Important! The server does not allow an Event's suspend and resume times to both fall in the spring-forward transition period. For example, if the suspend time is 2ND SUNDAY OF MARCH 2:20AM and the resume time is 2ND SUNDAY OF MARCH 2:50AM, the server resolves both times to 3:00 a.m., which is not allowed.
- execTrigger function
- Time dependencies
- Facility to test schedule criteria for the current year
- External job synchronization

Note: For the server to adjust the time, you must state the time in the schedule criteria explicitly (for example, 2:30 SUNDAY, but not EVERY 35 MINUTES).

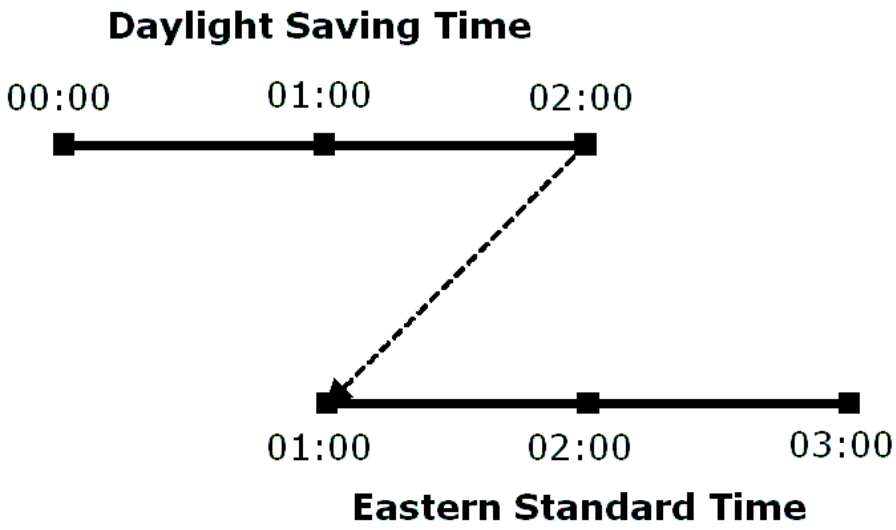
The server does not adjust the times for the following to 3:00 a.m. if they fall in the spring-forward transition period:

- Event simulation of jobs with time dependencies
- Event simulation for future years
- genTime function
- Event listing
- Forecasting
- Facility to test schedule criteria for future years

Note: When the schedule criteria takes the form EVERY [N] MINUTES, the server does not adjust the time for the spring-forward transition period. For example, if NOW is 00:00 a.m. on the day spring forward occurs, EVERY 35 MINUTES schedules the Event at 12:35 a.m., 1:10 a.m., 1:45 a.m., 3:15 a.m., 3:50 a.m., and so on. In this case, the 2:15 a.m. scheduled time occurs at 3:15 a.m., not at 3:00 a.m.

Fall-Backward Adjustments

In the North American Eastern Standard Time Zone, DST ends on the first Sunday of November at 2:00 a.m.; the time falls back one hour to 1:00 a.m. The following illustration displays DST fallback:



Events that are scheduled during the fall-back transition period (1:00 a.m. to 2:00 a.m. time period on the day fall back occurs) will not run twice. For example, on the day fall back occurs, 1:15 a.m. occurs twice spaced one hour apart. The first time, 1:15 a.m. occurs on the DST scale; the second time, 1:15 a.m. occurs on the EST scale. Events always run during the second time interval (Standard scale). In this case, the Event runs at 1:15 a.m. EST.

For example, at 12:40 a.m. on November 4, 2007, you define an Event with a schedule criteria of HOURLY. At 1:59:59 a.m. DST, the time falls back to 1:00 a.m. EST. The server triggers the Event at 1:00 a.m. EST, not one hour earlier at 1:00 a.m. DST.

Note: You can simulate your Applications and generate a forecast report for the day fall back occurs to find out which jobs run and when Events are scheduled on that day.

Appendix A: Using Scheduling Terms

This section contains the following topics:

Scheduling Terms (see page 657)

Scheduling Terms

You can use the following scheduling terms in CA Workload Automation DE:

ALL

Specifies all of the elements within a period (equivalent to ANY and EVERY). For example, you can use FIRST WORKDAY OF ALL MONTHS instead of FIRST WORKDAY OF MONTH.

AM

Specifies the time between 00:00 and up to, but not including, 12:00.

AND

Indicates an optional connecting word that you can use to specify multiple similar scheduling terms (days of the month, months of the year) and ordinal numbers, for example:

- FIRST AND LAST WORKDAY OF EACH MONTH
- TUESDAY AND WEDNESDAY
- JUNE, JULY AND AUGUST

Note: You cannot use this term with times of day.

ANY

Specifies all of the elements within a period (equivalent to ALL and EVERY), for example, ANY WORKDAY OF 6th MONTH OF YEAR.

ANYDAY

Indicates that no day of the week restriction exists. For example, ANYDAY OF JUN specifies all the days in June.

COMPLETE

Indicates that you want the schedule to occur only during a complete period. For example, you can specify FIRST COMPLETE WEEK OF YEAR. If a week starts in one year but finishes in the next year, CA Workload Automation DE interprets the expression to mean the next complete week.

DAILY

Specifies every one day.

DAYS

Specifies a period of time in days, for example, TODAY PLUS 5 DAYS.

EACH

Specifies every occurrence of a particular day or period, for example, EACH MONDAY OF OCTOBER.

ENDING

Specifies the ending point for a repeating schedule, for example, DAILY AT 9AM STARTING TOMORROW ENDING 1JUL2005. This term is identical to the term UNTIL.

Note: You cannot use this term in Run and Do not Run statements for jobs and Applications.

EVERY

Specifies all elements within a period (equivalent to ANY and EACH), for example, EVERY 1 HOUR.

Note: You cannot use this term as a run criteria for a job.

EVERY *n units*

Indicates how often, in units of time, you want the schedule to recur. The units can be seconds, minutes, hours, workdays, days, weeks, months, or years, for example, EVERY 5 MINUTES, EVERY 1 HOUR, EVERY 2 DAYS.

Note: You cannot use this term as a run criteria for a job.

EXCEPT

Indicates an exception. You can use different criteria as exceptions, for example:

- DAILY EXCEPT MONDAY TUESDAY
- FRIDAYS EXCEPT FIRST AND LAST DAY OF MONTH
- WEEKDAYS EXCEPT WEDNESDAYS
- HOLIDAYS EXCEPT CHRISTMAS
- WORKDAYS EXCEPT INVENTORY_DAY

Note: You cannot use this term in a run criteria for a job with a date, for example, Run DAILY EXCEPT OCT 3, 2008. However, you can accomplish this using a Do not run statement: Run DAILY; Do not run OCT 3, 2008.

FIRST

Specifies the first occurrence of a particular day or period, for example, FIRST MONDAY OF MONTH or FIRST DAY OF 2nd FISCAL_MONTH OF FISCAL_YEAR. You can also use the ordinal number 1ST, for example, 1ST HOLIDAY OF YEAR.

HOLIDAYS

Specifies all of the holidays in each calendar. You can refer to specific holidays by name, for example, CHRISTMAS.

HOURLY

Specifies every one hour, for example:

- HOURLY means 00:00, 01:00, and so on.
- HOURLY STARTING AT 2:15 means 2:15, 3:15, and so on.

LAST

Specifies the final occurrence of a particular day or period, for example:

- LAST HOLIDAY OF YEAR
- LAST WORKDAY OF FISCAL_MONTH
- FIRST AND LAST TUESDAY OF MONTH

LESS *n units*

Requests an adjustment by subtracting a value which must be a whole number. For example, you can specify LESS 2 WEEKDAYS, LESS 1 MONTH, or LESS 0 WORKDAYS. To obtain the second last day of the month, you can use LAST DAY OF MONTH LESS 1 DAY.

MIDDAY

Specifies the time 12:00 or noon.

MIDNIGHT

Specifies the time 24:00. CA Workload Automation DE recognizes 24:00 as the same as 00:00. For example, if you specify MIDNIGHT MONDAY, it is the same as 00:00 TUESDAY.

MINUTE/MINUTES

Specifies a period of time in minutes, for example, EVERY 2 MINUTES.

MONTH/MONTHS

Specifies a period of time in months, for example 1ST MONTH OF YEAR.

MONTHLY

Specifies every one month.

Note: You cannot use this qualifier by itself within a job definition. You can use RUN 15TH MONTHLY, but you cannot use RUN MONTHLY.

NOON

Specifies the time 12:00.

NOW

Specifies the current virtual time. It refers to the scheduled time of the Event that triggered the Application.

Note: You cannot use this term in an Event definition on its own. You can, however, specify a submission time for a job such as NOW PLUS 1 HOUR.

OF

Indicates a connecting word for scheduling terms, for example, LAST DAY OF MONTH or 1ST-9TH WORKDAY OF EACH YEAR.

ON

Indicates a connecting word indicating that a date follows, for example, 7PM ON THURSDAYS instead of 7PM THURSDAYS.

ONCE

Indicates that you want an Event to run once only.

OR

Indicates a connecting word that is used as a conjunction. This term lets you specify either of two similar scheduling terms. For example, 9PM ON FIRST MONDAY OR TUESDAY OF MONTH means that if the month starts on Tuesday, then this job will run on Tuesday. Otherwise, it will run on Monday.

Note: The term OR implies whichever instance occurs first and should only be used when it is meaningful. A statement such as 1ST JANUARY or 2ND FEBRUARY is meaningless to CA Workload Automation DE.

PLUS *n units*

Requests an adjustment by adding a value, which must be a whole number. For example, you can specify PLUS 2 WEEKDAYS, PLUS 0 WORKDAYS, or PLUS 1 WEEK. To obtain the first day after the last workday of the month, you can use LAST WORKDAY OF MONTH PLUS 1 DAY.

PM

Specifies a time between 12:00 and up to, but not including, 24:00. It identifies a preceding number as a time of day, for example, 6PM.

REALNOW

Specifies the actual time that the Event is triggered. For example, you can specify the start time for a job as REALNOW PLUS 1 HOUR.

ROUND

Indicates that the computed time is an integral multiple of the units parameter. You can use ROUND with EVERY *n* UNITS phrases, for example:

- EVERY 1 HOUR ROUND requests every hour at the hour mark.
- EVERY 6 HOURS ROUND requests the times 00:00, 06:00, 12:00, and 18:00.
- EVERY 5 HOURS ROUND provides the next hour that is an integral multiple of five hours starting at 00:00 on the current day (that is, 00:00 today, 05:00, 10:00, 15:00, 20:00, 1:00 tomorrow, 06:00, and so on).

STARTING

Specifies the starting point for a repeating schedule. You can use an actual date or time after STARTING, for example, DAILY STARTING NOV 6, HOURLY STARTING 2PM.

Note: You cannot use this term in Run and Do not Run statements for jobs and Applications.

THE

Indicates an optional connecting word, for example, 5TH DAY OF THE MONTH instead of 5TH DAY OF MONTH.

THIS

Specifies the current period, for example, 1ST WORKDAY OF THIS MONTH. You do not usually need the term THIS when you schedule jobs; however, it can be useful when you want to ensure CA Workload Automation DE generates date and time variables in the current period. For example, to simulate an Event for the last workday of this month plus one day, use LAST WORKDAY OF THIS MONTH PLUS 1 DAY.

TODAY

Specifies the current virtual or scheduled day.

TOMORROW

Specifies the day after today.

UNTIL

Specifies the ending point for a repeating schedule, for example, DAILY AT 9AM STARTING TOMORROW UNTIL 1JUL2005. This term is identical to the term ENDING.

Note: You cannot use this term in Run and Do not Run statements for jobs and Applications.

WEEK

Specifies a seven-day period. The first day of the week depends on your geographical location.

WEEKDAYS

Specifies Monday through Friday.

WEEKEND

Specifies a grouping of Saturday and Sunday together. For example, ANYDAY OF WEEKEND resolves to Saturday and Sunday.

WEEKLY

Specifies every one week.

WITHIN

Indicates a connecting word for period processing. For example, 5TH MONDAY WITHIN MONTH selects only the 5th Monday in the months that have five Mondays. CA Workload Automation DE ignores the other months.

WORKDAYS

Specifies all of the days worked in each calendar. Workdays exclude holidays.

YEARLY

Specifies every one year.

YESTERDAY

Specifies the day before today.

Index

%

%IF statement • 640
%VAR statement • 600

A

abandon predecessor dependencies • 575
abandon resource dependencies • 576
abandon variables dependencies • 577
agent groups
  defined • 58
  load balancing • 58
  running job on all agents in • 59
agents
  defined • 56
  load balancing • 58
  relationship with server • 56
  running jobs on group • 77, 78
Alert notifications
  defined • 609
  examples • 626
Alerts
  continuous monitoring • 64
  notifications • 609
anticipated end times
  defined • 19
Application integration tags • 25
Application parameters
  defined • 499
  exporting • 51
  removing from Event • 515
  specifying in Application • 22
  specifying in Event • 514
Application Services jobs, description • 95
Application templates
  defined • 21
  example • 24
  usage • 23
Applications
  adding comments • 45
  concurrent updates • 17
  copying SAP jobs into • 374
  critical path • 19
  default Event • 499
  defined • 13
  defining predecessor dependencies • 555
  delaying jobs from running until complete in previous generation • 47, 90
  deleting • 41
  dependencies between different • 552
  distant • 552
  downloading • 40
  editing multiple job definitions in • 37
  examples • 219
  generation phase • 15
  home • 552
  integration tags • 25
  locating jobs in • 36
  locking • 42
  mapping to business service • 52
  preventing concurrent generations • 45, 46, 511
  process phase • 15
  relationships • 547
  scheduling • 512
  setting Application defaults • 44
  setting up Alert notifications • 623
  setting up email notifications • 613
  setting up SNMP notifications • 630
  specifying JavaScript script • 637
  storing JavaScript scripts • 634
  subApplications • 18
  synchronizing • 219
  unlocking • 43
  versioning • 16
  zoom settings • 54
arguments, passing • 79
average execution time
  overriding • 91

B

business service, mapping Application to • 52
bypassing
  jobs • 578

C

CA WA jobs • 200
CA WA Restart Option
  enabling for job restarts • 490
  overriding statements • 492
  specifying statements • 491

CA Workload Automation DE
connector • 25, 52
commands
setting the reason option as mandatory • 50
comments
adding Application • 45
adding job • 85
conditional codes • 489
conditional jobs • 61
conditional schedule criteria
defined • 640
examples • 91, 569, 575, 578, 640
copy JCL
specifying • 488
Copy Single Request jobs, defining • 321
counts
suspend • 497
CPU Monitoring jobs, defining • 264
CPU usage, monitoring • 264
critical paths
defined • 19
enabling • 48

D

database
control concurrent access • 588
Database jobs
DB Monitor • 213
DB Stored Procedure • 203
DB Trigger • 210
description • 202
SQL • 216
Database Monitor Events
defined • 504
defining • 532
Database Trigger Events
defined • 504
defining • 535
Daylight Saving Time
defined • 653
fall backward • 656
spring forward • 654
DB Monitor jobs
defined • 209
defining • 213
DB Stored Procedure jobs
defining • 203
output variables • 68
DB Trigger jobs
defined • 209
defining • 210
defaults
Application • 44
job • 85
delaying job submission • 569
dependencies
global variable • 591
predecessor • 547
resource • 581
time • 562
Disk Monitoring jobs, defining • 269
disk space, monitoring • 269
distant Application • 552
downloading Applications • 40
dueout times
defined • 564
propagation • 565
reset • 565

E

email notifications
defined • 608
examples • 44, 573, 574
Entity Bean jobs
create an entity bean • 98
description • 97
remove an entity bean • 107
update an entity bean • 103
environment variables
OpenVMS • 448
specifying • 82
Tandem NSK • 450
UNIX • 461
Windows • 477
EVAR context • 22
Event triggers • 494
Events
Alerts, triggered by • 609
creating • 505
Database Monitor • 504
Database Trigger • 504
Daylight Saving Time, effect • 654, 656
defined • 493
expect times • 496
File Trigger • 500
JMS Subscribe • 502

prevent from triggering • 46, 511
priority • 498
recurring • 512
removing Application parameters • 515
resuming • 509
SAP Event Monitor • 503
scheduling • 512
specifying Application parameters • 514
specifying JavaScript scripts • 636
suppress no work notification • 49
suspending • 509
triggers • 494
Variable Dependency Monitor • 505
z/OS Data Set Trigger • 501
execution time
maximum • 567
minimum • 568
exit codes
defining • 80
examples • 626
expect times
defined • 496
exporting
Application parameters • 51
external jobs
defined • 552
defining • 219
description • 219
examples • 219

F

File Transfer jobs • 223
File Trigger Events
defined • 500
defining • 516
examples • 509, 516
File Trigger jobs
defined • 274
defining • 275
examples • 275
polling interval • 275
FTP client • 224
FTP jobs
defined • 223
defining • 226
examples • 226
SSL • 238
FTP server • 225

G

generations
concurrent • 18
delaying jobs from running until complete in previous • 47, 90
preventing from running at same time • 45, 46, 511
global variable dependencies
abandon time • 577
defined • 591
deleting • 600
job submission • 592
modifying • 597
setting up • 592
global variables
%VAR statement • 600

H

home Application • 552
HTTP jobs
define • 112
description • 111

I

i5/OS
naming conventions • 435
running UNIX workload • 435
i5/OS jobs
defining • 437
description • 435
passing keyword parameters • 444
SBMJOB options • 444
setting process priority • 445
using user's library list • 443
inheritance
defined • 553
override • 559
invoke a program over HTTP • 112
IP address, monitoring • 288
IP Monitoring jobs, defining • 288

J

JavaScript scripts
Alerts, run by • 609
Application definition • 637
common uses of • 633
Event definition • 636

examples • 626
job definition • 638
storing in Application • 634
JMS Publish jobs
define • 120
description • 117
JMS Subscribe Events
defined • 502
defining • 526
JMS Subscribe jobs
define • 124
description • 117
JMX jobs
define a JMX-MBean Attribute Get job • 130
define a JMX-MBean Attribute Set job • 133
define a JMX-MBean Create Instance job • 137
define a JMX-MBean Operation job • 141
define a JMX-MBean Remove Instance job • 145
define a JMX-MBean Subscribe job • 148
description • 128
job output variables • 65
job profiles
defined • 62
specifying • 93
jobs
abandoning predecessor dependencies • 575
adding comments • 85
anticipated end times • 19
average execution time • 91
bypassing • 578
CA WA • 200
conditional • 61
Copy Single Request • 321
CPU Monitoring • 264
critical • 92
Database • 202
DB Monitor • 213
DB Stored Procedure • 203
DB Trigger • 210
defaults • 85
defining exit codes • 80
delaying from running until complete in previous generation • 47, 90
delaying submission • 569
Disk Monitoring • 269
editing multiple definitions • 37
editing release conditions • 558
Entity Bean • 98, 103, 107
examples • 587, 626
execution time • 567, 568
external • 219
File Transfer • 223
File Trigger • 275
FTP • 226
hold count • 15
HTTP • 112
i5/OS • 437
IP Monitoring • 288
JMS Publish • 120
JMS Subscribe • 124
JMX • 128
JMX-MBean Attribute Get • 130
JMX-MBean Attribute Set • 133
JMX-MBean Create Instance • 137
JMX-MBean Operation • 141
JMX-MBean Remove Instance • 145
JMX-MBean Subscribe • 148
link • 201
locating in simulation • 508
locating on server • 73
Micro Focus • 256
Monitoring • 263
on-request • 60
OpenVMS • 446
Oracle E-Business Suite • 320
overdue • 563
overriding average execution time • 91
POJO • 155
premend • 568
priority • 582
Process Monitoring • 290
profiling • 62
Request Set • 324
resubmitting failed automatically • 87
RMI • 162
running on agent group • 77, 78
SAP • 362
SAP-Batch Input Session • 375
SAP-BW Info Package • 380
SAP-BW Process Chain • 382
SAP-Data Archiving • 384
SAP-Event • 387
SAP-Job Copy • 389
SAP-Process Monitor • 393
SAP-R3 • 396
SCP • 239
Session Bean • 167, 171
setting the command reason as mandatory • 50

setting up email notifications • 610
setting up SNMP notifications • 628
SFTP • 243
Single Request • 337
SNMP • 406
SNMP Subscribe • 407
SNMP Trap Send • 409
SNMP Value Get • 418
SNMP Value Set • 425
specifying JavaScript script • 638
specifying profile • 93
SQL • 216
subApplications, members of • 88
System • 434
Tandem NSK • 448
task • 202
Text File Reading and Monitoring • 293
UNIX • 452
Wake on LAN • 463
Web Service • 177
Windows • 468
Windows Event Log Monitoring • 301
Windows Service Monitoring • 306
z/OS • 477
z/OS-Data Set Trigger • 479
z/OS-Manual • 486
z/OS-Regular • 488

L

links
creating • 201
defined • 550
load balancing
defined • 58
running a job on an agent group • 77
locking Applications • 42

M

maximum execution time
defined • 567
notification • 573
specifing • 573
MBean, browsing • 153
MIB file, browsing • 415, 423, 431
MicroFocus jobs
defined • 255
defining • 256
minimum execution time
defined • 568
notification • 574
specify • 574
monitor states • 606
monitoring
continuously • 64
CPU usage • 264
disk space • 269
execution time • 573, 574
file activity • 275
IP address • 288
job's progress • 605
overdue jobs • 571, 572, 573
process status • 290
text file for text string • 293
Windows event logs • 301
Windows services • 306
Monitoring jobs
CPU Monitoring • 264
description • 263
Disk Monitoring • 269
IP Monitoring • 288
Process Monitoring • 290
Text File Reading and Monitoring • 293
Windows Event Log Monitoring • 301
Windows Service Monitoring • 306

N

namespace • 63
notifications
Alert • 609
defined • 605
email • 608
SNMP • 609
states • 606

O

on-request jobs • 60
OpenVMS jobs
defining • 446
environment variables • 448
Oracle Applications
application • 349
request set • 334
single request • 346
users • 335, 348
Oracle E-Business Suite jobs
Copy Single Request • 321

description • 320
Request Set • 324
Single Request • 337
overdue jobs
defined • 563

P

passing arguments • 79
payload consuming jobs • 71
payload producing jobs • 71
PeopleSoft jobs
description • 350
distribution • 357
POJO jobs
define • 155
description • 154
polling interval
File Trigger jobs • 275
predecessor dependencies
abandoning time • 575
defined • 547
defining • 555
inheritance • 553
manual • 551
overriding inheritance • 559
release conditions • 556, 558
simplifying • 550
predecessors
defined • 13
release conditions • 556, 558
premend jobs • 568
priority
Event • 498
job • 582
process
monitoring status • 290
priority, UNIX • 457
Process Monitoring jobs, defining • 290
propagation of dueout times • 565

R

release conditions
predecessor and successor dependencies • 558
predecessor dependency • 556
Remote Execution jobs
defining • 359
description • 358
renewable resources
examples • 587, 588
Request Set jobs, defining • 324
request sets
running • 324
searching for • 334
searching for users to notify upon completion • 335
reserving resources • 586
resources
abandoning time • 576
priority • 582
reserving • 586
setting up dependencies • 584
resubmitting failed jobs automatically • 87
resume times • 497
resuming Events, scheduled • 509
retry count • 87
RMI jobs
define • 162
description • 161
run frequency
specifying • 75
workload objects • 56

S

SAP Event Monitor Events
defined • 503
defining • 529
SAP events
monitoring on CA Workload Automation DE • 387
triggering on CA Workload Automation DE • 387
triggering workload when raised • 529
SAP jobs
copying into Application • 374
defined • 362
filtering and listing on SAP system • 364
SAP processes, monitoring • 393
SAP system
copying jobs into Application • 374
filtering and listing jobs • 364
setting up connection • 363
SAP-Batch Input Session jobs
defining • 375
filtering and listing on SAP system • 365
SAP-BW Info Package jobs
defining • 380
filtering and listing on SAP system • 367

SAP-BW Process Chain jobs
defining • 382
filtering and listing on SAP system • 369
SAP-Data Archiving jobs
defining • 384
SAP-Event jobs, defining • 387
SAP-Job Copy jobs
defining • 389
SAP-Process Monitor jobs
defining • 393
SAP-R3 jobs
defining • 396
filtering and listing on SAP system • 371
schedule criteria
conditional • 640
Daylight Saving Time • 653
days of the week and months • 645
defined • 644
Event • 495
examples • 512, 645, 648, 650, 651
simulation • 496
suspend/resume times • 497
time dependencies • 562
time of day • 650
time zones • 651
SCP jobs, defining • 239
secure file transfers • 239, 243
server
loacting a job • 73
relationship with agents • 56
Session Bean jobs
defined • 165
defining • 167, 171
SFTP jobs, defining • 243
shells, UNIX jobs • 457
Single Request jobs, defining • 337
single requests
copying • 321
running • 337
searching for • 346
searching for users to notify upon completion • 348
SNMP jobs • 406
SNMP notifications • 609
SNMP Subscribe jobs, defining • 407
SNMP Trap Send jobs
browsing MIB file • 415
defining • 409
SNMP Value Get jobs
browsing MIB file • 423
defining • 418
output variables • 69
SNMP Value Set jobs
browsing MIB file • 431
defining • 425
output variables • 69
SQL jobs
defining • 216
output variables • 67
SSL FTP usage • 238
stored procedures • 203
subApplications
defined • 18
defining • 88
waiting for previous generation • 88
submission times
conditional schedule criteria • 569
Daylight Saving Time, effect • 654
defined • 563
delaying • 569
successor dependencies
defining • 555
release conditions • 558
successors
defined • 13
release upon condition • 556, 558
suppress no work notification • 49
suspend counts • 497
suspend times • 497
suspending Events, scheduled • 509
synchronizing Applications • 219
System jobs • 434

T

Tandem NSK jobs
defining • 448
environment variables • 450
tasks
creating • 202
defined • 551
Text File Reading and Monitoring jobs, defining • 293
threshold resources
examples • 589
time dependencies
abandoning predecessor • 575
abandoning resource • 576

abandoning variable • 577
bypassing jobs • 578
defined • 562
delaying job submission • 569
marking job overdue • 571, 572, 573
time format and position, Text File Reading and Monitoring jobs • 299
time zones • 651
trigger Event
database activity, based on • 532, 535
file activity, based on • 516
global variable expression, when met • 542
inherit trigger user • 505
JMS message, when received • 526
SAP event, when raised • 529
suppress no work notification • 49
z/OS data set activity, based on • 519

U

UNIX jobs
defining • 452
description • 451
environment variables • 461
modifying resource limits • 458
script/command name specifications • 456
setting process priority • 457
shells, how selected • 457
unlocking Applications • 43
usage • 64
user with different passwords, defining • 63
users
Oracle Applications • 335, 348

V

variable dependencies
abandon time • 577
defined • 591
deleting • 600
job submission • 592
modifying • 597
setting up • 592
Variable Dependency Monitor Events
defined • 505
defining • 542
verifying
defining • 479
specifying CA WA Restart Option statements • 491
zoom settings, Application graphs • 54
versioning, Application • 16

W

Wake on LAN jobs
defining • 463
description • 462
Web Service jobs
define • 177
Windows Event Log Monitoring jobs • 301
Windows event logs, monitoring • 301
Windows job objects
defined • 474
defining • 475
Windows jobs
command file specifications • 474
defining • 468
description • 467
environment variables • 477
Windows Service Monitoring jobs, defining • 306
Windows services, monitoring • 306
workload
scheduling • 512
trigger based on file activity • 516
trigger based on z/OS data set activity • 519
trigger when global variable expression is met • 542
trigger when JMS message received • 526
trigger when number of table rows changes • 532
trigger when SAP event raised • 529
trigger when table rows are updated • 535
workload objects • 56

Z

z/OS Data Set Trigger Events
defined • 501
defining • 519
examples • 519
z/OS jobs
condition codes • 489
description • 477
enabling CA WA Restart Option for job restarts • 490
overriding CA WA Restart Option statements • 492
specifying CA WA Restart Option statements • 491

z/OS-Data Set Trigger jobs
    defined • 478
    defining • 479
z/OS-Manual jobs, defining • 486
z/OS-Regular jobs, defining • 488
zoom settings, Application graphs • 54