

Unicenter® AutoSys® Job Management

Core Components

Instructor Workbook

UA100



Computer Associates®
EDK2UAJ45CIE

– PROPRIETARY AND CONFIDENTIAL INFORMATION –

These education materials and the related computer software program (hereinafter referred to as the "Education Materials") are for the end user's informational purposes only and are subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

These Education Materials may not be copied, transferred, reproduced, disclosed or distributed, in whole or in part, without the prior written consent of CA. These Education Materials are proprietary information and a trade secret of CA. Title to these Education Materials remains with CA, and these Education Materials are protected by the copyright laws of the United States and international treaties. All authorized reproductions must be marked with this legend.

RESTRICTED RIGHTS LEGEND

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 THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

© 2003 Computer Associates International, Inc. (CA). CA confidential and proprietary information for CA internal use only. No unauthorized copying or distribution permitted.

All rights reserved.

All trademarks, trade names, service marks or logos referenced herein belong to their respective companies.

Call Computer Associates technical services for any information not covered in this manual or the related publications. In North America, see your Computer Associates Product Support Directory for the appropriate telephone number to call for direct support, or you may call 1-800-645-3042 or 631-342-4683 and your call will be returned as soon as possible.

Outside North America, contact your local Computer Associates technical support center for assistance.

Table of Contents



Introduction

Welcome	viii
About This Course	xii
Case Study: RBC, Inc.	xiv
Unicenter AutoSys JM Architecture	xv

1 • Manage Jobs

Create and Run Command Jobs	1-3
Job Definitions	1-4
Job Statuses	1-6
The Event Processor	1-8
Create and Run File Watcher Jobs	1-14
Example	1-14
Create and Run Box Jobs	1-20
Example	1-21
View Processing of Events	1-28
Set Alarms	1-30
Features	1-30
What are Alarms?	1-30

2 • Manage Events

Send a Job On Hold Event	2-3
Scheduler Console in Windows	2-5
Job Activity Console in UNIX	2-6
Send a Job On Ice Event	2-10
Send a Kill Job Event	2-13
Send a Change Status Event	2-16
Discuss	2-17
Send a Force Start Job Event	2-19
Example	2-19

Send a Comment Event	2-21
Send a Set Global Event	2-23
Examples	2-23

3 • Use the Scheduler or Job Activity Console

Create Filters	3-3
Examples	3-3
Sort Columns in the Console	3-8

4 • Use Basic Commands

View a Job Detail Report for Multiple Jobs	4-3
Syntax	4-3
Start the Command Line Interface	4-7
View Chase Output for All Running Jobs	4-9
Syntax	4-10
Verify that the Event Processor and Event Server are Running	4-12
Syntax	4-12
View Current Version of Unicenter AutoSys JM	4-15
Syntax	4-15
Report Job Dependencies	4-17
Syntax	4-17
Example	4-19
View the Event Processor Log or Remote Agent Log	4-21
Syntax	4-21
Example	4-22
Send Events	4-23
Syntax	4-23
Examples	4-25

5 • Schedule Jobs

Create and Test Jobs Dependent on Job Status	5-3
Syntax	5-3
Success Dependencies	5-4
Failure Dependencies	5-11
Terminated Dependencies	5-14
Not Running Dependencies	5-17
Done Dependencies	5-19

Exit Code Dependencies	5-21
Use the And, Or, and () Operators	5-23
Submit a Date/Time Schedule	5-27
Submit Alarms/Terminators	5-32
Alarms	5-32
Terminators	5-33
Specify Additional Job Attributes	5-36

6 • Use Calendars

Create Calendars	6-3
Modify Calendars	6-6
Apply Rules to Calendars	6-7
Examples	6-8
Import and Export Calendars	6-10
Schedule Jobs Using Calendars	6-13

7 • Create Job Monitors and Browsers

Create a Job Monitor	7-3
Examples	7-3
Create a Browser Report	7-9
Examples	7-9

8 • Use the Web Interface

Set Up and Customize the Web Interface	8-3
Manage Jobs	8-9
Generate Remote Access Reports	8-13
Run Simulations Using Xpert	8-16

A • Skill Builder Solutions



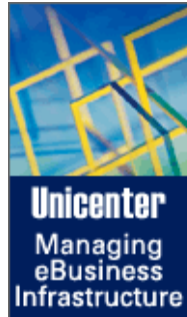
Introduction

Welcome

Slide 1



Target Audience



Welcome to *Unicenter® AutoSys® Job Management (Unicenter AutoSys JM): Core Components* training by Computer Associates. This course was specifically designed for System Operators, as well as anyone interested in learning foundational skills for managing jobs using Unicenter AutoSys JM. In your job, you are responsible for scheduling and managing jobs according to requirements that change constantly throughout your enterprise. As your employer continues to increase the size and complexity of the business, your job management responsibilities are growing exponentially.

Learning Path

Slide 2



This course will show you how to define and manage jobs to support the distributed operations that exist across your corporation with a minimum of manual intervention. You will learn how to use the features and functions of Unicenter AutoSys JM Version 4.5 to automate the often massive processing of jobs your organization requires.

To learn about additional training solutions designed for your job role or this software product, visit:

gems.ca.com/Gemsmarketing/CourseFinder.asp

Here you will find links to the Course Catalog, Learning Paths, Registration information, and Schedules. Learning Paths will help you determine the best training combination to enhance job performance, learn advanced skills, or become certified.



Instructor
Notes

- | | |
|----------------|---|
| Prepare | <p>Ensure that all the Windows User IDs are entered through autosys_secure before starting module 1 of this course.</p> <p>Check the Instructor Prep document prior to beginning class. If one was not provided for you with your class materials, you may download it from Edweb or from the server: \\usiledev\Developement\AutoSys Development-UA100 Core Concepts\Instructor.</p> |
| Orient | <p>Show slide 1. Welcome students to class.</p> <p>Briefly describe your background and qualifications. Invite students to share their backgrounds and experiences as System Operators. State facility policies such as break times, smoking areas, rest room locations.</p> <p>Show slide 2.</p> <p>Click the sample graphic to open your live Internet Browser. Demonstrate GEMS navigation and show students how to find Learning Paths, course catalogs, registration information, and schedules.</p> |

About This Workbook

Each task in this course is presented using the following instructional events, which follow a specific sequence.



Instructional Event

Description

Interactive Demonstration

You will follow along on your student computer as your instructor demonstrates each step in a task. Your instructor will use this opportunity to explain the concepts, the conditions that drive which options should be selected, and other supporting information essential for you to understand the task's purpose. Interactive Demonstrations are marked by the icons at the left, a silhouette of an instructor indicating an overhead screen while a student works at a computer running the indicated operating system.

Skill Practice

You will practice a task just demonstrated, while your instructor provides coaching. Skill Practices are marked by the icon at the left, a silhouette of an instructor coaching a student working at a computer.

Skill Builder

You will build confidence applying the task just learned to real-world business problems written around a fictional scenario, so that you understand not just *how* to perform the task, but *why*. Skill Builders are marked by the icon at the left, a silhouette of a student working at his or her own pace to solve the problem.



Instructor
Notes

Explain

the types of exercises used throughout the course.

Notify

Tell students that Skill Practices may not be provided for every task, especially when the task is quite simple - it is too redundant.

Tell students that Skill Builders are typically end-of-module assessments that combine several tasks into a single problem, to more effectively simulate the real world.

Tell students that depending on the task's complexity, we may skip demonstrations and go straight to Skill Practices at instructor discretion, and invite them to tell you when they feel comfortable with this practice.

Conventions

The following conventions are used throughout your Student Workbook.

Convention	Use	Example
Bold	GUI elements; or anything you must click.	Click OK to continue. Select jobname .
<i>Italics</i>	New terms, placeholder text, and emphasis.	A <i>dialog box</i> is a window that appears to collect information from the user. Enter a range from 0 to <i>n</i> , where <i>n</i> equals the number of tapes. Do <i>not</i> erase the backup tapes.
Andale	Commands that should be typed exactly as written. Job names	At the prompt, type: c:\Program Files\myfolder name_jobname
<i>Andale Italics</i>	A placeholder where you substitute a real value for the italicized text. In Job naming convention, use your own name and then the job name, separated by an underscore.	At the prompt, type: c:\Program Files\ <i>folderName</i> <i>name_jobname</i> or john_job1
SMALL CAPS	Names of keys.	To reboot, press CTRL+ALT+DEL
ALL CAPS	Types of status	SUCCESS, TERMINATED, and so on.



Instructor
Notes

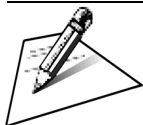
Emphasize

that the naming convention described above is for class purposes only; students should ask if there is a similar convention in place once they return to work.

We suggest writing a sample job name on the white board, using either your name or the name of one your students.

We Welcome Your Ideas

The Computer Associates Education Team is committed to providing you, our student, with the most effective training experience possible. To help us make sure these guides and our courses reflect real-world work situations, we encourage you to visit ca.com/education/forms/improvements.htm to share your successes or challenges working with our software since taking this course. We will use your suggestions to revise subsequent courses.



Instructor
Notes

-
- Ask** students to either inform you of any errors they find, or to report them to us using the form found at the URL provided in their Workbooks.
- Ask** students to also suggest their ideas for improving the course beyond fixing errors, such as more descriptive business problems we might use for Skill Builder problems in later course editions.

About This Course

Slide 3



This course is *performance-based* training. It was designed to teach only the tasks you will likely perform within 90 to 120 days of completion. This is why we use the term *tasks* throughout the course materials, rather than *lessons*. Where new terms, features, or functions of the Unicenter AutoSys JM software must be understood before you can perform the task, they will be introduced within the context of proper task performance, to facilitate your ability to recall this information later. To do this, we will introduce a fictional case study to simulate business conditions.

Course Length

2 days

Prerequisite Skills

- Knowledge of your corporate job management policies.
- Basic understanding of Windows and UNIX is helpful.



Instructor
Notes

Explain	what we mean by performance-based training.
Slide 3	the course's duration, in days.
Ask	students if they all possess the prerequisite skills listed in their Workbooks.
Slide 4	Cover the course agenda.

Course Agenda

Slide 4

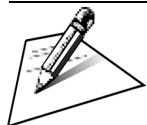


Day 1

- Introduction
 - Welcome
 - About This Course
 - Case Study: RBC, Inc.
- Module 1: Manage Jobs
- Module 2: Manage Events
- Module 3: Use the Scheduler or Job Activity Console
- Module 4: Use Basic Commands

Day 2

- Module 5: Schedule Jobs
- Module 6: Use Calendars
- Module 7: Create Job Monitors and Browsers
- Module 8: Use the Web Interface



Slide 4 Cover the course agenda.

Instructor
Notes

Case Study: RBC, Inc.

The Really Big Corporation (RBC, Inc.) is a fictional entity that represents any large business and will be used throughout this course to illustrate task performance. Like most large enterprises, its growth is due to a number of mergers and acquisitions. Today, the conglomerate has business holdings in many diverse markets. Legacy systems and data integration present numerous challenges to RBC. With offices and subsidiaries scattered world-wide, RBC is continually looking for ways to streamline its operations.

Job Management at RBC

Throughout this course, you will be asked to play the role of a key stakeholder at RBC, to solve real-world business problems using Unicenter AutoSys JM.

To keep pace with a faster-than-expected growth rate, your operations team at RBC needs event-driven scheduling, centralized monitoring, and automated error-recovery for all the platforms currently in use, as well as any to be added later.

RBC, Inc. has offices in London, Hong Kong, and Sydney in addition to its U.S.-based headquarters in Chicago. You need to manage jobs across multiple time zones, with a minimal amount of manual intervention. You also need to automate the various holiday schedules and other schedules in which jobs in specific countries can and cannot run. Since your Sales operation is tightly integrated with Marketing, Production, and Inventory, you need to coordinate the processing of any jobs that may depend on other jobs across these departments to maximize efficiency.



Instructor
Notes

Unicenter AutoSys JM Architecture

You can define, run, and manage Unicenter AutoSys JM instances and jobs using a graphical user, command line, or web interface. This course presents task performance using only the graphical user interface, but some basic commands are covered in Module 4; the full command line and web interfaces are presented in the next course offered in the Learning Path, *UA200: Unicenter AutoSys Job Management Advanced Job Scheduling*. Unicenter AutoSys JM features a UNIX® and Windows® NT® (and higher) graphical user interface (GUI).

What is a Job?

At work, you will likely find that people tend to use the term *job* to mean different things at different times. *Job* could represent the Unicenter AutoSys JM job, the remote agent job, or the application job that the Unicenter AutoSys JM job is attempting to run.

<i>Unicenter AutoSys JM Job:</i>	A definition you create that instructs the system what command, executable, or batch file to run.
<i>Remote Agent Job:</i>	A service (Windows) or process (UNIX) that runs on a client machine, and handles communication between that client machine and Unicenter AutoSys JM.
<i>Application Job:</i>	The command, executable, or batch file that is started by the remote agent.

As you progress through this course, the distinctions will be clearer.

Note • In Unicenter AutoSys JM 4.0, the Job name may be no more than 30 characters in length. However, in Unicenter AutoSys JM 4.5 the Job name may be more than 80 characters in length. No spaces are permitted, but you may use dashes, underscores, and periods.



Instructor
Notes

Clarify the distinction among an Autosys job, an application job, and a remote agent job.

What is an Instance?

The term *instance* is used to represent a licensed version of Unicenter AutoSys JM and at least one client machine running a unique Remote Agent. In reality, your operation is likely to have dozens of Remote Agents communicating concurrently with the same database. Any status changes or service requests posted to the database are called *Events*.

These terms will be clarified in an example to be presented in the next section, and in Module 1.

The following three main components comprise an instance:

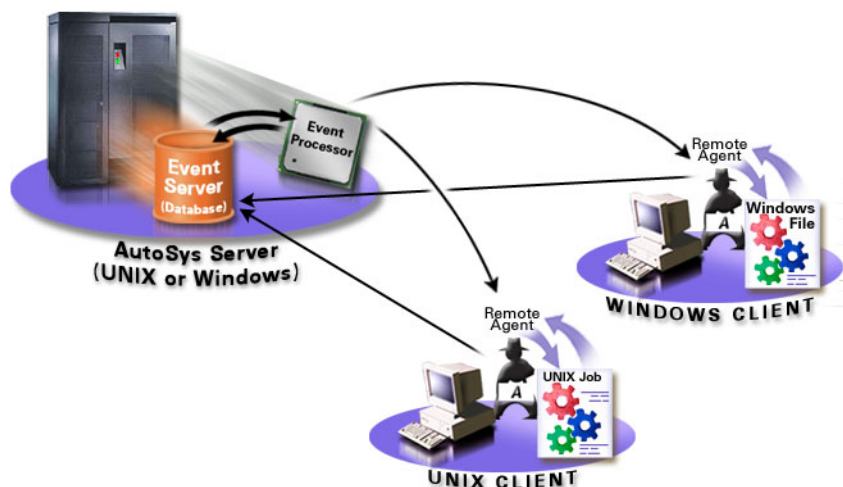
Component	Description
Event Server	Stores all events and Unicenter AutoSys JM job information (relational database).
Event Processor	Interprets events. Evaluates Unicenter AutoSys JM job definitions and notifies a Remote Agent to initiate actions.
Remote Agent	<p>Performs tasks; sends resulting job statuses back to the Event Server.</p> <p>UNIX: a remote agent is a temporary process started by the Event Processor to perform a specific task on a remote (client) machine.</p> <p>Windows: a remote agent is a Windows service running on a remote (client) machine that is directed by the Event Processor to perform specific tasks.</p>



Instructor
Notes

Clarify the distinction among an Autosys job, an application job, and a remote agent job.

Slide 5



This graphic represents the typical installation architecture of Unicenter AutoSys JM.

The Event Processor and the Event Server typically reside on the same machine, but can be installed on separate machines. The Event Processor (Unicenter AutoSys JM engine) scans the Event Server (Unicenter AutoSys JM database) approximately every half-second for the next event to process.

When the Event Processor encounters an event that is ready, it starts to process it; for example, a STARTJOB event.

Note • The following graphic will be animated during class, to facilitate your comprehension of the process about to be explained.



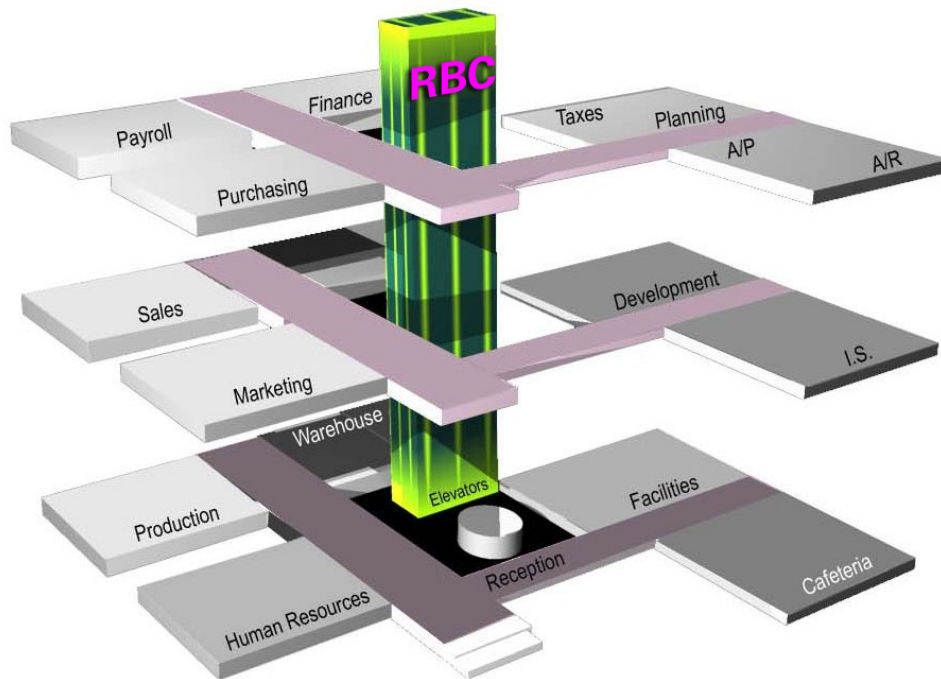
Instructor
Notes

Slide 5 Discuss the typical architecture of UAJM.

Slide 6



Processing a STARTJOB event involves the following sequence:



- 1 The Event Processor notifies the Event Server (database) that the job is now STARTING. (In this course, we use all CAPS to represent the various event statuses. A complete list of statuses is provided for you in *Module 1, Manage Jobs.*)



Instructor
Notes

Slide 6

WARNING

To start the animation, click the Advance button in the lower right corner of your slide.

AVOID THE ENTER KEY! This will kick you out of the presentation.

Explain what the Event Processor and Event Server do as the animation progresses.

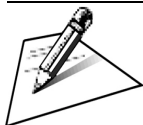
When you get to the job attributes, emphasize that the Event Processor determines if all of the starting conditions have been met. If so, it sends the STARTING event to the database for that job.

Emphasize that what is shown in the animation is a single job, and is actually happening all over the network, for every event the EP finds.

- 2 The Event Processor contacts the client machine that was specified in the Job Definition and establishes a connection. The Event Processor starts a Remote Agent job on that machine and passes all of the Unicenter AutoSys JM job's attributes to it. Job attributes are stored in the database as Job Definitions, which contain such information as when to start, what machine to run, and under what conditions.
- 3 The Remote Agent sends back an acknowledgment to the Event Processor that it has received all of the job's parameters. The connection between the Event Processor and the Remote Agent is terminated. The Event Processor resumes its scan of the Event Server for the next event to process.
- 4 The Remote Agent tells the operating system to execute the command defined in the job and establishes a new connection; this time, to the Event Server (database). The Remote Agent tells the Event Server that the job is now RUNNING and then disconnects. Simultaneously, the Remote Agent starts a Remote Agent Log that records this single job's progress. Meanwhile, the Event Processor now encounters the new event, RUNNING, processes it, and records it in its own log called the Event Processor Log. Processing includes starting any other jobs that may depend on the current status of this job to start.

It is in this manner that you can use Unicenter AutoSys JM to manage complex jobstreams and to automate the processing of thousands of jobs.

- 5 The job completes and the Remote Agent receives its exit code. The Remote Agent evaluates whether the job ran successfully.
- 6 The Remote Agent again connects to the Event Server to report another status change, this time to report the job's terminal status as SUCCESS or FAILURE, and then disconnects. As long as the job status is SUCCESS, the Remote Agent will delete its log file.
- 7 The Remote Agent then terminates.



Instructor
Notes

The Event Processor, which has been scanning the Event Server every half second for events to process, receives the job's completion status and now begins to evaluate if any other jobs depend on the successful completion of the one just processed. If so, those jobs will now begin.

The basic architecture just described allows you to centralize the processing of thousands of jobs typically generated by a large business and provides the foundation on which the remainder of this course is built.



Instructor
Notes



Manage Jobs

Module Objectives

Slide 1-1



After this module, you will be able to:

- Create and Run Command Jobs
- Create and Run File Watcher Jobs
- Create and Run Box Jobs
- View Processing of Events
- Set Alarms

Module Overview

The Sales Organization at RBC regularly runs the following jobs: a job that runs a batch file on each team member's sales for the time specified and a job that watches for the output file produced by the first job. Since these are not the only jobs for which you are responsible, you need a way to automate processing so that all scheduled jobs run with a minimum of manual intervention.

In this module, you will learn how to manage basic job details including executing commands, watching for files, and nesting similar jobs to optimize resource usage. In addition, you will learn how to view the events as each job is processed.



Slide 1-1 Use the slide to state the learning objectives for this Module.

Relate: Ask one or two students to discuss jobs they may need to automate at their work sites similar to the situation described in the Module Overview.

Instructor
Notes

Slide 1-2



Task 1: Create and Run Command Jobs

Term	Definition
<i>Job</i>	<p>A single action that can be performed on a valid client machine. Depending upon the platform, a job could be a single command, shell script, executable, or batch file.</p> <p>There are three types of Unicenter AutoSys JM jobs: Command, File Watcher, and Box.</p> <p>Note • In the Introduction, we provided a broader definition of the term <i>job</i>. The definition provided here is for a Unicenter AutoSys JM job.</p>
<i>Command Job</i>	Command Jobs execute commands.
<i>Jobstream</i>	A flow of jobs linked by starting dates/times or conditions.

Slide 1-3



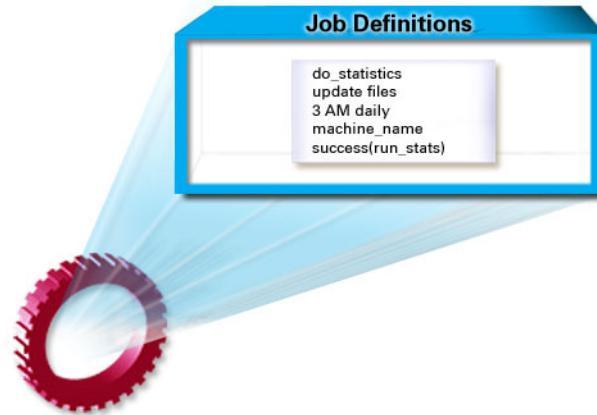
In the Unicenter AutoSys JM environment, a *job* is a single action that can be performed on a valid client machine. In UNIX, this action can be any single command or shell script, and in Windows, this action can be any single command, executable, or batch file. All activity controlled by Unicenter AutoSys JM is based on the job; like the way a gear runs an entire machine, the job is the basic building block on which the entire automated operations cycle is built.

Instructor
Notes

- Slide 1-2** Use the slide to introduce Task 1.
- Slide 1-3** Use the slide to introduce the concept of a job. Explain how jobs can be linked together to form jobstreams.
- Define** job and command job. Also, clarify the distinction among a Unicenter AutoSys JM job, an application job, and a remote job.
- Discuss** the benefits of using both UNIX and Windows. Point out the strengths and weaknesses between the two operating systems with regards to task in Unicenter AutoSys JM.

Job Definitions

Slide 1-4



Each job definition contains a variety of qualifying attributes, including the conditions specifying *when*, *where*, and *if* a job should be run.

Using utilities, you can define a job by assigning it a name and specifying the attributes that describe its associated behavior. These specifications make up the job definition.

Jobs are defined using the *Job Definition* tool in UNIX and the *Job Editor* tool in Windows. Alternatively, jobs can also be defined by executing Job Information Language (JIL) commands using a command line.

This course presents use of the graphical user interface only; command line and JIL are covered in *UA200: Unicenter AutoSys Job Management Advanced Job Scheduling*.



Slide 1-4 Use the graphic to help explain job definitions. Clarify the use and meaning of each attribute. Compare and contrast defining jobs in UNIX and Windows.
Point out that some jobs may never run; they could be defined to recuperate from an error situation.

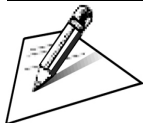
Instructor
Notes

Job definitions include the following set of qualifying attributes: when, if, and where. To effectively manage jobs, you will learn to combine these various starting parameters:

<i>When</i>	specific start dates or times recurrences, such as every Friday calendars, such as company holidays a watched file becomes available
<i>If</i>	a trigger job returns a SUCCESS or other required status a watched file becomes available
<i>Where</i>	a machine name; the name of the client machine on which the job is to run

The following table lists the basic file definition attributes:

Attribute	Description
Job Name	The unique job identifier by which a job is referenced. There is a maximum of 80 characters (no spaces) for a job name.
Machine Name	The name of the machine on which the command is to be executed.
Command	The command the job is to execute.



Instructor
Notes

Compare and contrast each of the file definition parameters; try to give examples. If you have some Best Practices to share for devising effective job names back at work, please share them.

Job Statuses

At any given point in time, the Event Server will display one of the following statuses for a specified job. The statuses are organized logically in the order in which they will likely be encountered:

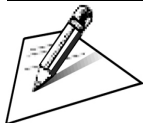
Status	Description
INACTIVE	The job has not yet been processed.
STARTING	The Event Processor initiated the start job procedure with the Remote Agent.
RUNNING	<p>The job is running.</p> <p>If the job is a box job, this means the jobs within the box may have been started.</p> <p>If the job is a command or file watcher job, this means that the process is actually running on the remote machine.</p>
SUCCESS	The job exited and reported an exit code equal to or less than the <i>maximum exit code for success</i> . The default exit code is 0 and is interpreted as SUCCESS.
FAILURE	<p>The job exited with an exit code greater than the <i>maximum exit code for success</i>.</p> <p>If the job is a box job, this means one job in the box exited with a FAILURE status or that the Exit Condition for Box Failure was true.</p>



Instructor
Notes

Discuss	how exit codes are determined and processed.
Refer	students to the following sources for more information: <i>Chapter 3</i> , Unicenter AutoSys JM <i>User Guide V4.5</i>
Remind Students	...as you saw in the animation, the Event Processor is always looking for the next event to process. As a single job progresses through the processing cycle, each of its statuses could trigger the start of subsequent jobs, which is why this product is capable of complex jobstream automation.

Status	Description
TERMINATED	<p>The job was terminated while in the RUNNING state. This means a user sent a KILLJOB event, or the job was set to terminate if the box holding it failed.</p> <p>However, if the job inside a box fails, the status is FAILURE, not TERMINATED.</p> <p>Jobs could also be terminated if they exceed the maximum run time, or if they were killed from the command line with a UNIX kill command.</p>
RESTART	Hardware or application problems prevented the job from starting, so the job had to be rescheduled to restart.
QUE_WAIT	The job can logically run, but there are insufficient physical resources currently available.
ACTIVATED	The job itself has not yet started, but the box it is in is now RUNNING. This status is only applicable to box jobs.
ON_HOLD	The job is on hold and cannot be run until it receives a JOB_OFF_HOLD event.
ON_ICE	“More than just on hold.” This means the job has been removed from all conditions and logic, but is still defined. It means it has been deactivated. Jobs will remain ON_ICE until they receive the JOB_OFF_ICE event.



Instructor
Notes

- Ask** students to discuss the difference between ON_HOLD and ON_ICE.
- (When an On Hold job is taken off hold, the On Hold job will be scheduled to run if the starting conditions were met, while an On Ice job that is taken Off Ice will not run, even when the On Ice job starting conditions are met.)
- Another distinction is that On Hold jobs will obstruct all jobs scheduled to run downstream of it. On Ice jobs will have no effect on jobs scheduled downstream of it.

The Event Processor

The *Event Processor* is the Unicenter AutoSys JM engine. When the Event Processor is not running, you cannot initiate new actions. The Event Processor interprets and processes all the events it reads from the database. Sometimes called the `event_daemon`, the Event Processor is the program running as a UNIX process or as a Windows service that actually runs Unicenter AutoSys JM. The Event Processor schedules and starts jobs. After it is started, the Event Processor scans the database every half-second for events to be processed. When the Event Processor finds one, it checks whether the event satisfies the starting conditions for other jobs in the database. Based on this information, the Event Processor first determines what actions are to be taken, then instructs the appropriate remote agent process to perform the actions. These actions may be the starting or stopping of jobs, checking for resources, monitoring existing jobs, or initiating corrective procedures.

What is a Command Job?

The *command job* is the most common type of job. The command can be a shell script, an executable program, a file transfer, and so forth. When you run this type of job, you will execute the specified command on a client machine. When all the starting conditions are met, Unicenter AutoSys JM runs this command and captures its exit code upon completion. The exit event (SUCCESS or FAILURE) and the exit code value are stored in the database. Exit codes will be covered in greater detail in *Module 5, Schedule Jobs*.

For all job names, use this convention: *name_[u|w]_jobname*, where *name* represents your name, *u* represents UNIX and *w* represents Windows.

Interactive Demonstration

Task Purpose: Create a basic command job in UNIX.

- 1 From the command prompt, launch the Unicenter AutoSys JM toolbar using the command: `autosc`



Instructor
Notes

Performing the Demonstration

- | | |
|--|---|
| <p>Ask</p> | <p>how to verify the Event Processor is running? <code>chk_auto_up</code> command</p> <p>Mention that it will be covered in Module 4.</p> |
| <p>Performing the Demonstration</p> | <p>As you go through the steps of the Interactive Demonstration, discuss each step to clarify why you are making certain selections and not others. For example, you select Command for a command job. Explain that you will cover Box and File Watcher in subsequent tasks. When you get to Date/Time Options, explain why you are NOT selecting them right now, and so on. When you get to Execute on Machine, spend a minute or two discussing what possibilities exist and how Unicenter AutoSys JM processes each.</p> |

- 2 From the Unicenter AutoSys JM toolbar, click **Job Definition**. The Job Definition tool opens.

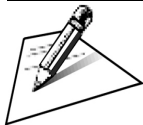
The screenshot shows the 'Job Definition' dialog box with the following fields and options:

- Buttons:** Clear, Delete, Save, Adv Features, Exit
- Job Name:** Text field with a Search button.
- Job Type:** Dropdown menu with options: Box, Command (selected), File Watcher.
- Edit OneTime Over-Rides ?** Radio buttons: Yes, No.
- Name of Box this Job is IN:** Text field with a Search button.
- Owner:** Text field containing 'autosys@dcauto1'.
- Description:** Text field.
- Starting Parameters:**
 - Is the Start Date/Time Dependent ?** Radio buttons: Yes, No.
 - Date / Time Options ...** Button.
 - Starting Condition:** Text field.
- Command & File Watch Information:**
 - Execute On Machine:** Text field.
 - Command To Execute:** Text field.
 - File To Watch for...:** Text field.

- 3 In the **Job Name** field, type:

`name_u_command1`

where *name* represents your first or last name.



Instructor
Notes

■ Manage Jobs

Interactive Demonstration

- 4 Verify the **Job Type** is **Command**. If not, select **Command**.
- 5 In the **Execute on Machine** field, type your *machinename*.

Note • Your instructor will supply the appropriate machine name.

- 6 In the **Command To Execute** field, type:
echo "hello"

Note • You cannot redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Adv. Features, as follows:

/tmp/*name_u_command1.out*

- 7 Click **Save**.
- 8 Click **Exit**.

Interactive Demonstration

Task Purpose: Run the previously created job in UNIX.

- 1 From the Unicenter AutoSys JM toolbar, click **Ops Console**. The Job Activity Console opens.
- 2 Choose **View ▶ Select Jobs**. The Job Selection tool opens.
- 3 For **Job Selection**, select **All Jobs**.
- 4 For **Sort Order**, sort by **Job Name**.
- 5 Click **Apply** and then click **OK**.

In the Job Activity Console, a list of jobs is displayed.



Instructor
Notes

Introduce	the Adv. Features tab (UNIX) and its associated options.
Provide	the value, or the way to determine the value, students should type in place of <i>machinename</i> .

- 6 Verify that **Freeze Frame** in the **Show** group at the bottom of the Job Activity Console is *not* enabled. If it is, click to disable it.
- 7 In the **Reports** group of the Job Activity Console, select **Event**.
- 8 Highlight the job name you created in the previous exercise.
- 9 In the **Actions** area (at the bottom of the screen) click **Start Job**.
- 10 Click **Yes** to start the job.

Watch the **Event Report** area for updates to the status of the job you specified.

Interactive Demonstration



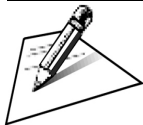
Task Purpose: Create a basic command job in Windows. In this exercise, you will use the *echo* command as an example.

- 1 Launch the Graphical Interface: Click **Start** ▶ **Program Files** ▶ **Computer Associates** ▶ **Unicenter** ▶ **AutoSys Job Management (W45)** ▶ **Graphical Interface**.

Note • The Graphical Interface is commonly referred to as the Unicenter AutoSys JM toolbar, or informally called the *button bar*.

- 2 From the Unicenter AutoSys JM toolbar, click **Job Editor**.
- 3 From the **Job Editor** dialog, choose **File** ▶ **New**. The **New Job** dialog opens.
- 4 Name the job as follows:

`name_w_command1`
- 5 Verify that the **Instance** name, W45, is correct.
- 6 From the **Job Type** list, select **Command**.



Instructor
Notes

7 Click **OK**. The New Job dialog closes.

8 In the **Command** field, type:

```
echo hello
```

Note • You cannot redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Command Info, as follows:

```
c:\temp\name_w_command1.out
```

9 Next to the **Send To Machine** field, click **Add**.

10 In the New Machine Name dialog, type your *machinename* and click **OK**.

11 Choose **File ► Save**.

12 Click **OK** to clear the message that notifies you the job definition was successfully saved.

13 Exit the Job Editor.

Interactive Demonstration

Task Purpose: Run a command job in Windows. In this exercise, you will learn how to manually start and run the job created in the previous exercise.

1 From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.

Note • If this is the first time you have run Scheduler Console, you will be prompted to create a default filter. Click **Yes**.

2 Right-click your **command1** job and choose **Start Job** from the shortcut menu.



Instructor
Notes

Introduce the **Command Info** tab and its associated options.

Prepare **IAD -Run Command Job in Windows** - before you are able to have the job run successfully, autosys_secure utility must be run and the administrator must be set up as a superuser with password.

- 3 Verify that you want to start the job by clicking **Yes** to the message.
- 4 Click **Refresh**. You will now see a status of **SUCCESS** in the Status column for that job.
- 5 To observe job progress, click **Job Detail Report** while your job is selected.
- 6 From the drop-down list, choose **Event**.
- 7 Click **OK**.



Skill Practice

Task Purpose: Practice creating and running a basic command job in Windows or UNIX. Follow the same naming convention presented previously.

- 1 Create a command job called `calc1`.
- 2 In Windows, set your job to run the system calculator *or* in UNIX, run the command: `ls`
- 3 Run `calc1` on your *machinename*.
- 4 Check the status of `calc1` and discuss your results with the class.

Task Summary

You learned that a *job* is the basic building block in Unicenter AutoSys JM. You practiced creating and running basic command jobs in UNIX and Windows. In the next Task, you will practice creating and running basic File Watcher jobs. Eventually, you will learn how to apply these basic tasks to automatically manage complex jobstreams.



Instructor
Notes

Review Skill Practice

Permit time for students to work on their own; coach those that seem to struggle. After they complete the task, ask questions to check their understanding.

When all students have completed the Skill Practice, have them discuss the difficulties and decisions they encountered.

Briefly explain the Job Detail Report.

Close the Task by reviewing concepts and checking students' understanding.

Slide 1-5



Task 2: Create and Run File Watcher Jobs

Term

Definition

File Watcher

A File Watcher job is similar to a command job, except it watches for the arrival of a specified file, and signals the file's arrival.

Slide 1-6



Where command jobs start the specified command, all *File Watcher* jobs do is start a process that monitors for the existence and size of a specific operating system file. When that file reaches a certain minimum size and has stopped growing in size, the File Watcher job completes successfully, indicating that the file has "arrived."

By creating jobs dependent on File Watcher jobs, you can take events that are outside of Unicenter AutoSys JM and integrate them into its job processing.

Example

In the previous task, you learned that RBC's Sales Organization regularly runs a job that executes a batch file on each team member's sales, and another job that watches for the output file produced after the first job executes. Consider that this output file is normally produced after 2 AM, when you have minimal staff scheduled. You could create a File Watcher job to start at the anticipated time to wait for the output file to be posted. You could also create two command jobs to be dependent upon the File Watcher job: one that depends on its success to start, and the other that depends on its failure, adding further automation to the process. Job dependencies will be discussed in *Module 5, Schedule Jobs*.



Instructor
Notes

Slide 1-5 Use this slide to introduce Task 2.

Slide 1-6 Use the graphic on this slide to help you explain the concept of a file watcher job. Contrast with command and box jobs.

The basic File Watcher job definition has the following required attributes:

Attribute	Description
<i>Job Name</i>	The unique job identifier by which a job is referenced.
<i>File Name to Watch For</i>	The name of the file to watch. You must specify the full path including the name.
<i>Machine Name</i>	The name of the machine where the file being watched is expected.

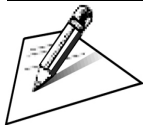


Interactive Demonstration

Task Purpose: Create a basic File Watcher job in UNIX. In this exercise, you will create a job definition that watches for the file created in the previous job:

`name_u_command1.out`

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**. The Job Definition tool opens.
- 2 In the **Job Name** field, type:
`name_u_filewatcher1`
- 3 Select a **Job Type** of **File Watcher**.
- 4 In the **Execute on Machine** field, type the name of your machine.
- 5 In the **File to Watch for** field, type:
`/tmp/name_u_command1.out`



Instructor
Notes

Perform the Interactive Demo

As you make each selection, briefly tell students why you are making that decision instead of another.

Add value: briefly include the conditions under which an option NOT being demonstrated at the moment might otherwise be used or needed. Remind students to take notes as you lecture.

■ Manage Jobs

Interactive Demonstration

- 6 Click **Save**.
- 7 Exit the Job Definition tool.
- 8 To run your **filewatcher1** job, switch to the Unicenter AutoSys JM toolbar, and click **Ops Console**. The Job Activity Console opens.
- 9 Locate the job:
 - a Choose **View ▶ Select Jobs**.
 - b Select **All Jobs**.
 - c Click **Apply**.
 - d Click **OK**. The Job Activity Console now displays a list of available jobs.
- 10 Highlight **name_u_filewatcher1**. At the bottom of the Job Activity Console, in the **Show** group, ensure that **Freeze Frame** is not enabled.
- 11 In the **Reports** group, select **Event**.
- 12 In the **Actions** group, click **Start Job**.
- 13 Click **Yes** to clear the message box and start the job you specified.

Watch the Event Report area for updates to the status of the job you specified.

Interactive Demonstration

Task Purpose: Create a basic File Watcher job in Windows.

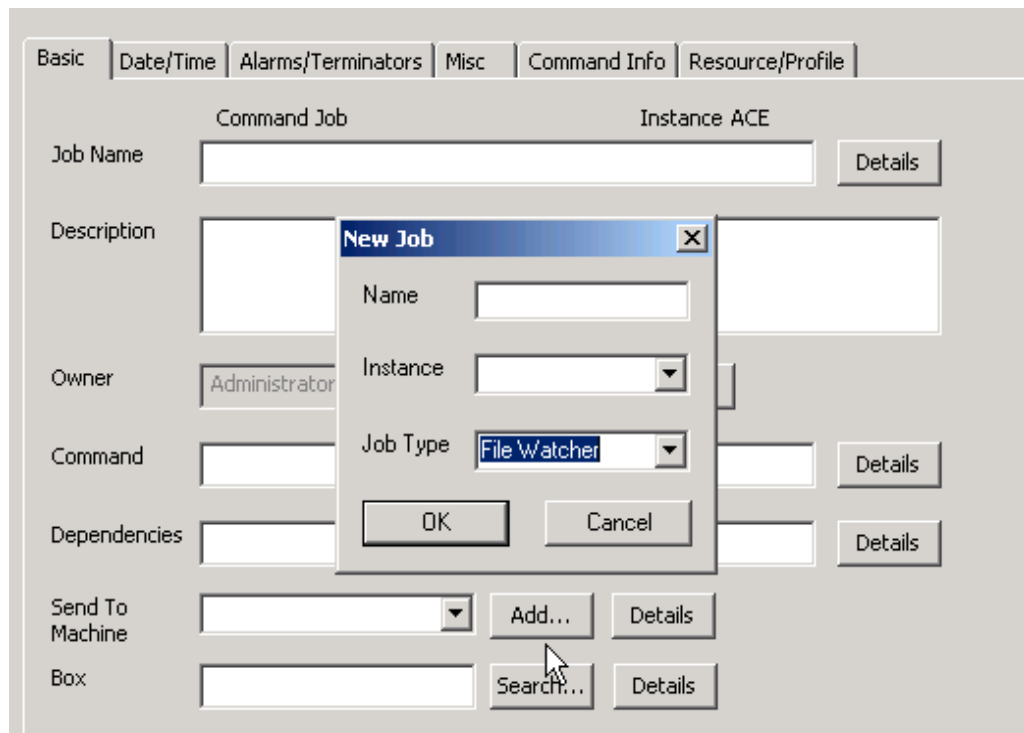
- 1 From the Unicenter AutoSys JM toolbar, click **Job Editor**.



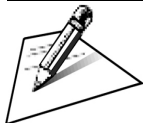
Instructor
Notes

Add You may wish to briefly describe what **Freeze Frame** is and why we keep disabling it throughout this module.

- 2 Choose File ► New.



- 3 In the **Name** field, type:
`name_w_filewatcher1`
- 4 Verify the instance name is correct.
- 5 From the **Job Type** list, select **File Watcher**.
- 6 Click **OK**. The Job Editor displays the File Watcher job dialog box.



Instructor
Notes

- 7 On the **Basic** tab, in the **File to Watch** field, type:
`c:\temp\name_w_command1.out`
- 8 In the **Send to Machine** field, select your *machinename*.
- 9 Choose **File ► Save**. Verify the instance name is correct and click **OK**.
- 10 Exit the Job Editor.
- 11 From the **Scheduler Console**, click **Refresh**. The job **filewatcher1** should now appear in the list.



Interactive Demonstration

Task Purpose: Run the job called **filewatcher1** in Windows.

- 1 From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.
- 2 Right-click the job created in the previous exercise: **filewatcher1** and choose **Start Job** from the shortcut menu.
- 3 Click **Yes** to dismiss the Console dialog.
- 4 After you submit the job, click **Refresh**.
- 5 Click **Job Detail Report** to see the job's progress.
- 6 You will now see the Status column in the Scheduler Console update to **SUCCESS** for this job.



Instructor
Notes



Skill Practice

Task Purpose: Practice creating and running a basic File Watcher job in UNIX or Windows.

- 1 Create a new File Watcher job using the same naming conventions presented throughout this module and call it `filewatcher2`.
- 2 Set your job to watch for a file called `echo2.txt`.
- 3 Run `filewatcher2` on your *machinename*.
- 4 Check the status of `filewatcher2` and discuss your results with the class.

Task Summary

You learned how you can use Unicenter AutoSys JM to watch for required files as they become available, extending the product's automation capability. In the next task, you will learn how to group jobs with similar starting conditions.



Instructor
Notes

Skill Practice

When performing the Skill Practice, walk around the classroom coaching students.

Ask students why the job has not completed? Since `echo2.txt` has not been created, the file watcher is still running. If any student should need assistance in creating the file, please take a moment to cover the steps.

Close

the task by reviewing key concepts and checking students' understanding.

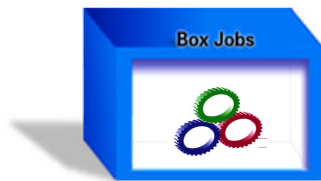
Slide 1-7



Task 3: Create and Run Box Jobs

Term	Definition
<i>Box</i>	Box jobs are containers that hold jobs related by start time or date.

Slide 1-8



A *box job* contains jobs with similar starting conditions. In the previous Task, you learned about file watcher jobs. Assume you have a series of jobs that all depend on the same watched file to start. You can add the jobs to a box and then make only the box job dependent on the watched file, instead of having to set up the same dependency for each job in the jobstream.

Since a box is actually a container of jobs with similar starting conditions, such as date/time conditions, or job dependency conditions, you use boxes to group those jobs according to scheduling parameters, and not as an organizational method.

Attribute	Description
Box Name	The unique identifier by which the box is referenced.



Instructor
Notes

Slide 1-7 Use the slide to introduce Task 3.

Slide 1-8 Use the graphic to explain the concept of a box job.

Example

Examine the following list of jobs:

Job	Parameters
Accounting_Daily	Runs each morning at 7 AM Eastern time
Accounting_Close	Runs each day at 5 PM Eastern time
Accounting_Report	Runs when a file called <code>sales.txt</code> is available
Sales_Products	Runs each morning at 7 AM Eastern time
Sales_Region	Runs quarterly
Sales_Rep	Runs when a file called <code>sales.txt</code> is available
Sales_Accounts	Runs daily, but cannot be run at the same time Accounting_Daily or Sales_Products is run

Discussion As a class or in small groups, discuss which jobs could be placed in a box and which ones should not.



Instructor
Notes

Discuss use the example provided as a class discussion. Split the class into small groups or simply invite comments.

Example Solution *Jobs that run daily at the same time* could be placed in a box. However, *jobs that process accounts* is insufficient information to qualify for a box. Under such a category, you may likely have jobs that process accounts quarterly, monthly, or only upon exception. Such jobs should not be placed in the same box, because they lack similar starting conditions.

Name or list on the white board the jobs that can be placed in a box.

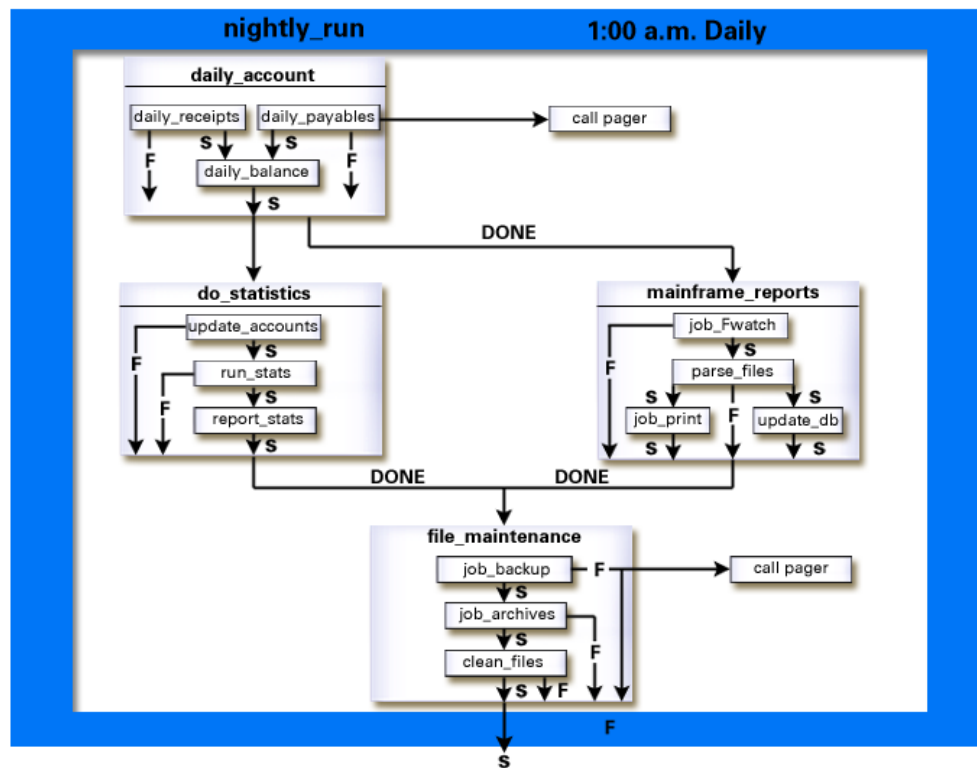
■ Manage Jobs

Example

Slide 1-9



Boxes can be nested; that is, you can place a box within a box, as the following illustration shows.



The arrows marked with an S represent SUCCESS statuses. The job called **daily_balance** starts only when both **daily_receipts** and **daily_payables** are successful. If one of the jobs fail, **daily_balance** will not start and the box called **daily_account** will still show a status of running until someone investigates the problem.

Discussion

When the **daily_account** box finishes, one action will definitely happen and another action may happen if the box job was successful. Can you identify those actions?



Slide 1-9

Invite students to explain the jobstream pictured on the slide.

do_statistics will be executed only if **daily_account** is successful.
mainframe_reports will be executed when **daily_account** finishes running, regardless if it failed or succeeded.

Instructor
Notes

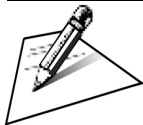


Interactive Demonstration

Task Purpose: Create a basic box job in UNIX.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type: `name_u_box1`
- 3 For the **Job Type**, select **Box**.
- 4 Click **Save**. The box job is now saved to the database, but it contains no jobs.
- 5 Add jobs to the box using the **Job Definition** tool:
 - a In the **Job Name** field, type the percent sign (%) and then click **Search** to display all defined jobs.
 - b Double-click to select the job called `name_u_command1`, created during a previous demonstration. The Job Definition is updated.
 - c In the **Name of Box this Job is In** field, type:
`name_u_box1`

Or, you may type the percent sign (%) and click **Search** for a list of all available box jobs.
 - d Click **Save**. The Job Definition is cleared.
 - e Repeat steps a through d for the job called `name_u_filewatcher1`.
- 6 Close the Job Definition.



Instructor
Notes



Interactive Demonstration

Task Purpose: Create a basic box job in Windows.

- 1 From the **Job Editor** dialog box, choose **File ► New**. The New Job dialog opens.
- 2 In the **Job Name** field, type:
`name_w_box1`
- 3 Verify the Instance name is correct.
- 4 In the **Job Type** field, select **Box**.
- 5 Click **OK**. The New Job dialog closes.
- 6 Choose **File ► Save**. Verify the Instance name is correct and click **OK**.
- 7 Exit the **Job Editor**.
- 8 Add your **command1** job to the Box.
 - a From the **Job Editor**, choose **File ► Open**.
 - b Verify the Instance name is correct.
 - c Click **Search** for a list of available jobs. If you type *name%* and then click **Search**, you will filter all defined jobs to display only those beginning with your name.
 - d Select your **command1** job and click **OK**.
 - e Next to the **Box** field (at the bottom of the Job Editor dialog box), click **Search**.
 - f In the **Box Selection** field, click **Search** for a list of available boxes. You can use the *name%* wildcard to filter results, just as you did in **step c**.



Instructor
Notes

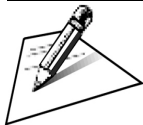
- g Select **box1** and click **OK**.
 - h Choose **File ▶ Save**.
- 9 Add your **filewatcher1** job to the Box.
 - a Choose **File ▶ Open**.
 - b Verify the instance name is correct.
 - c Click **Search** for a list of available jobs. Filter the displayed results as you did to add the previous job.
 - d Select your **filewatcher1** job and click **OK**.
 - e Next to the **Box** field, click **Search**.
 - f In the **Box Selection** field, click **Search** for a list of available boxes. Filter the results as you did to add the previous job.
 - g Select your **box1** job and click **OK**.
 - h Choose **File ▶ Save**.
- 10 Exit the Job Editor.

Interactive Demonstration



Task Purpose: Run a basic box job in UNIX. In this exercise, you will run **box1**, which was created in a previous demonstration.

- 1 Click **Ops Console** to open the Job Activity Console.
- 2 Choose **View ▶ Select Jobs**.
- 3 Confirm that **All Jobs** is selected; click **Apply** and then **OK**. A list of available jobs is displayed.



Instructor
Notes

- 4 Highlight your **box1** job.
- 5 In the **Show** group, disable **Freeze Frame**, and in the **Reports** group, select **Summary**.
- 6 Click **Start Job**.
- 7 Click **Yes** to clear the message box and start the job you specified.

Watch the Summary Report area for updates to the status of the box job you started.



Interactive Demonstration

Task Purpose: Run a basic box job in Windows. In this exercise, you will run *name_w_box1*, created in a previous demonstration.

- 1 Open the **Scheduler Console**.
- 2 Right-click your **box1** job and choose **Start Job**.
- 3 Click **Yes** at the Console dialog to confirm.
- 4 After you have submitted the job, click **Refresh**.

Note • You will now see a status of **RUNNING** for the job, **box1**. All jobs inside the box are activated. The box will wait until all jobs have completed before going to a **FINAL** status.

- 5 From the Scheduler Console, click **Job Detail Report** to see a list of events for all three jobs. The events should show **STARTING**, **RUNNING**, and **SUCCESS** in the detail report window.



Instructor
Notes

In the UNIX Demo: When you get to Step 5: Explain why you selected **Summary** instead of **Event**. Briefly discuss the difference.



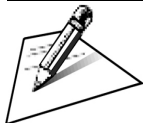
Skill Practice

Task Purpose: Practice creating and running box jobs in Windows or UNIX.

- 1 Create a new box job and continue the naming convention presented throughout this module: *name_[u|w]_box2*.
- 2 Add your jobs called **calc1** and **filewatcher2** to the box.
- 3 Run the box job and discuss your results.

Task Summary

You learned how to group jobs with similar starting conditions into boxes to facilitate processing. You have now practiced creating and running the three basic types of Unicenter AutoSys JM jobs, forming the foundation for the remainder of this course. In the next task, you will learn about another basic element of job automation: the event.



Wrap up

Have the students discuss what they learned in this task and how they will or will not be able to apply this at their work sites, by inviting them to suggest an operation they wish to automate; confirm their understanding by checking their logic.

Instructor
Notes

Slide 1-10



Task 4: View Processing of Events

Although the focus of this course is graphical interface usage, there are times when knowing certain commands could be useful.

The Event Processor Log contains a timestamped history of each event that occurs. Viewing this log is an alternative to monitoring “all jobs” and “all events.” To do so, use the `autosyslog` command:

Option	Description
-e	Indicates that the Event Processor Log is to be monitored. (When in this mode, in order to terminate the command, you must press CTRL+C.)
-J job_name	Indicates that the Remote Agent log for the specified <i>job_name</i> is to be viewed. You must issue this command on the machine where <i>job_name</i> ran. Otherwise, the following message will display: *** No remote agent files found for <i>job_name</i> ***
-p	Indicates the profile information for the job being viewed. Can only be used with the -J option.



Slide Use slide 1-10 just to introduce Task 4.

Instructor
Notes



Demonstration

Task Purpose: View the processing of events in Windows.

Note • To view the Event Processor log, you must execute this command on the machine that is running the Event Processor, or on a machine that can access the same \$AUTOUSER file system.

- 1 Click Start ► Programs ► Computer Associates ► Unicenter ► AutoSys Job Management (W45) ► Command Prompt (W45).
- 2 At the command prompt, type the following and then press ENTER:

```
autosyslog -e
```



Demonstration

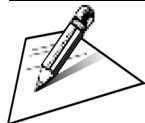
Task Purpose: View the processing of events in UNIX.

At the \$ prompt, type the following and press ENTER:

```
autosyslog -e
```

Task Summary

You now know how to view the processing of events, a feature useful for verifying jobs were processed as expected, or for troubleshooting purposes. Another feature essential for troubleshooting is the Alarm, the subject of the following task.



Instructor
Notes

Close the Task by reviewing concepts and checking for understanding.

Slide 1-11



Task 5: Set Alarms

You can set alarms in both Windows and UNIX. The Alarm feature is useful for troubleshooting job errors.

Features

Term	Definition
Alarm	A special event that is informational only and notifies you about situations requiring attention. Featured in both UNIX and Windows.
Alarm Sentry	Watches each instance of Unicenter AutoSys JM for alarms and displays a color change. Available only in Windows.
Alarm Manager	Lets you view, acknowledge, and close alarms. Available only in Windows.

What are Alarms?

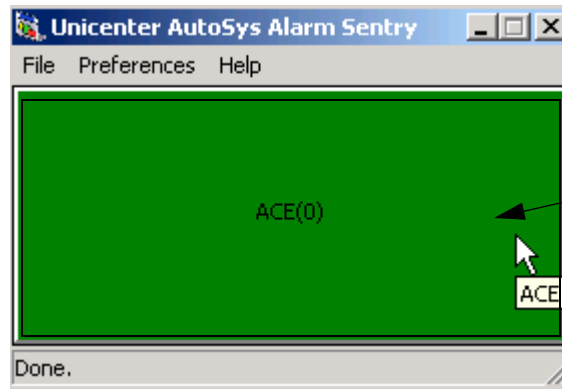
Alarms signal situations requiring your attention and are informational only. Alarms are sent through the system as an event. Jobs can be scheduled to run based on a number of conditions, but some thought is required to address incidents that require manual intervention. For example, a set of jobs could be dependent on the arrival of a file, and the file is long overdue. It is important that someone investigate the situation, make a decision, and resolve the problem.



Slide Use the slide to introduce Task 5. There are no more slides for this task.

Instructor
Notes

In Windows, you manage alarms using the Alarm Sentry and the Alarm Manager. To start the Alarm Sentry, click **Alarm Sentry** from the Unicenter AutoSys JM toolbar in Windows. The Alarm Sentry window is displayed:

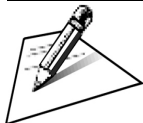


The entire field labeled ACE(0) is actually a *button*.

This graphic shows a single button because only one instance was defined.

The entire field labeled ACE(0) is what you click to open the Alarm Manager; the Alarm Sentry displays one such button for each Unicenter AutoSys JM instance you have installed. The number in parentheses that follows the instance name represents the number of open alarms for that instance. The color of the button (green) represents no new alarms were sent. When an alarm occurs, the button will change color (red) and the number will increase. The color will only change back to green after you have viewed — not necessarily acknowledged or closed — the alarms in the Alarm Manager.

Click the instance-specific button to open the Alarm Manager, which will also be instance-specific, or one that displays the alarms for this instance only. For the remainder of this course, we will refer to this colored field as the *Alarm Button*.



Instructor
Notes

NOTE

In some classroom environments, the Alarm button will *not* turn Red, as previously stated. Due to student machine color configurations, the button will turn black instead.

Note • In Windows, you can also open the Alarm Manager by clicking the Alarm Manager button from the toolbar, or by choosing Alarm Manager from the Applications menu of the Scheduler Console.

The Alarm Sentry and the Alarm Manager must be running to manage the alarms you defined using the Job Editor.

In UNIX, the Job Activity Console provides a similar button at the bottom right corner. However, this button is labeled *Alarm*, and does not feature the name of the instance. You click this button to open the Alarm Manager, which lists the alarm details for the job that produced an alarm. In other words, this single button functions as both Alarm Sentry and Alarm Manager. The color change (green to red) indicates new alarms were received since the last time you viewed alarms.

The **Advanced Features** button on the UNIX Job Definition accesses the parameters in the following table. In Windows, the same parameters are found on the Alarms/Terminators tab of the Job Editor.

	Parameter	Description
Alarms	Maximum Run Time	The job should not take any longer than the time specified; if it does, you may ask to be notified. You may also use the Terminate this job option to automatically terminate the job, as well as notify you.
	Minimum Run Time	The job should not end prior to the time specified; if it does, the alarm notifies someone to investigate.
	Send Alarm if this job fails	An alarm is sent whenever the specified job returns a status of FAILURE or TERMINATED.



Instructor
Notes

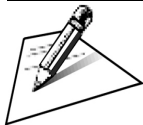
	Parameter	Description
Terminators	If this job fails, terminate the box it is in	Specifies whether the box that contains the job should be terminated if the job fails.
	If the box fails, terminate this job	Specifies whether the job should be terminated if the box containing it fails.
	Terminate this job __ minutes after starting	Automatically terminates any job that runs longer than the time specified.



Interactive Demonstration

Task Purpose: Create a UNIX job that fails and generates an alarm. For this demonstration, you will use the *echo* command, with an intentional syntax error, to produce the FAILURE status that will produce the alarm.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type:
`name_u_commandfail`
- 3 Select a **Job Type of Command**.
- 4 In the **Execute on Machine** field, type your *machinename*.



Instructor
Notes

- 5 In the **Command to Execute** field, type the following command with the syntax error as shown:

```
eco hello
```

Note • You cannot redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Adv. Features, as follows:

```
/tmp/name_u_commandfail.out
```

- 6 Click **Adv Features**.
- 7 Select **Yes** next to **Send Alarm if this job fails**.
- 8 Click **Save&Dismiss**.
- 9 Click **Exit**.
- 10 Run *name_u_command*fail:
- a From the Unicenter AutoSys JM toolbar, click **Ops Console**.
 - b Choose **View ▶ Select Jobs**.
 - c Select **All Jobs** and click **Apply** followed by **OK**.
 - d Under the **Show** group, disable **Freeze Frame**.
 - e Under the **Reports** group, select **Event**.
 - f Click **Start Job**. Click **Yes** to the confirm and clear the dialog.
- 11 Watch the Event Report area. You should note a **JOBFAILURE** status. Directly under that is the **Alarm** button, which should now be red to indicate the job failed.



STEP 11 NOTE

In some classroom environments, the Alarm button will *not* turn Red, as previously stated. Due to student machine color configurations, the button will turn black instead.

Instructor
Notes

12 View and acknowledge the failed job:

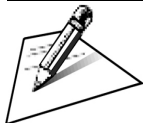
- a Click the red **Alarm** button to open the **Alarm Manager**. You will see a Job Failure alarm for the job that just failed, with a state of **Open**. Highlight this alarm. The state of *Open* indicates that the alarm has not been acknowledged or closed.
- b Select **Acknowledge** and click **Apply**. Notice the state has been changed to **Acknowledged**. You acknowledged **ONLY** the alarm for this job and no others.
- c Click **OK** to close the Alarm Manager. The Alarm button has returned to its default color of green.



Interactive Demonstration

Task Purpose: Create a Windows job that fails and generates an alarm. For this exercise, you will use the *echo* command, with an intentional syntax error, to produce the FAILURE status.

- 1 Start the Alarm Sentry, if it is not already running, by clicking **Alarm Sentry** from the Unicenter AutoSys JM toolbar.
- 2 Return to the **Job Editor**, and choose **File ► New**.
- 3 In the **Job Name** field, type:
`name_w_commandfail`
- 4 Verify that the instance name is correct.
- 5 From the **Job Type** list, select **Command**.
- 6 Click **OK**.



Instructor
Notes

Note • The misspelling of “echo” in the following step is intentional.

- 7 In the **Command** field, type:

eco hello

Note • You cannot redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Command Info, as follows:

c:\temp\name_w_commandfail.out

- 8 From the **Send To Machine** list, select your *machinename*.
- 9 Click the **Alarms/Terminators** tab and verify that the **Send Alarm if this job fails** option is selected.
- 10 Choose **File ► Save**. Verify the instance name is correct and click **OK**.
- 11 Exit the Job Editor.
- 12 Run the job just defined:
- a From the **Scheduler Console**, right-click *name_w_commandfail*.
 - b Choose **Start Job**.
 - c Click **Yes** at the Console dialog to confirm.
 - d After you have submitted the job, click **Refresh**. You will now see a status of **FAILURE** in the Status column. Note that the instance-specific alarm button has turned red due to the failure, and the number in parentheses updated.



Instructor
Notes

13 View and Acknowledge the Alarm:

- a Return to the Alarm Sentry you opened previously.

Note • The Alarm Sentry is red. This indicates that there are new alarms.

- b Click the instance-specific Alarm button. This opens the Alarm Manager.
- c Highlight **Alarm JOBFAILURE**.

You will see a Job failure alarm for the job that just failed, with a state of **Open**. This indicates that the alarm has not been acknowledged, or closed.

- d Select **Acknowledge** for the Alarm State and click **Apply**.

Notice the state has been changed to Acknowledged.

- e Select **Closed** for the Alarm State and click **Apply**. The alarm disappears.



Skill Practice

Task Purpose: Practice creating and running a job in UNIX or Windows that fails and generates an alarm. Use an intentional syntax error in a basic command or type a nonsensical command to force the failure condition.

- 1 Create a new job using the same naming convention used throughout this module and call it: `commandfail2`
- 2 Set the job to execute this nonsensical command, typed exactly as shown, to force the job to fail: `drxm echo.out`
- 3 Save and run the job.
- 4 View the alarm and discuss your results.



Instructor
Notes

Skill Builder: Manage Jobs



Business Problem

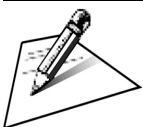
The Sales Team at Really Big Corporation uses a specific report on a regular basis, a sales report that lists the sales by product for the period specified. This report is called `sales_report.txt` and is generated by running a batch file called `sales_report.bat`. You need to know when the `sales_report.txt` file is available. You know you must create at least two jobs:

Job	Purpose
Use <code>name_[u w]_joba</code> as the job name.	to run the command that produces the report
Use <code>name_[u w]_jobb</code> as the job name.	to watch for the sales report file

How would you create and manage these jobs?

Hint

Consider creating at least one more job to help you solve this problem.



Instructor
Notes

Skill Builder Solution

Students should create jobA to run the command `sales_report.bat`, and jobB to watch for a file called `sales_report.txt`. jobB should give a status of successful, when the file it is watching becomes available. They should create a third job, a box job that contains jobA and jobB.

Verification

Files will be found at `c:\TEMP\`

Wrap Up

Make sure that all students understand the business problem and what they are supposed to accomplish.

Let them try to solve the problem on their own.

Coach students only when they struggle.

Once they complete the problem, discuss their solutions and any difficulties encountered.

Slide 1-12

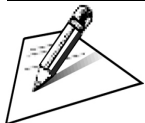


Module Summary

You should now be able to:

- Create and Run Command Jobs
- Create and Run File Watcher Jobs
- Create and Run Box Jobs
- View Processing of Events
- Set Alarms

This concludes job management core components. You now know how to use Windows and UNIX to create basic command, file watcher, and box jobs, as well as how to confirm their success and view their processing. With experience, you will likely find many other ways to correctly define and implement jobs. In the next module, you will learn more about events, giving you more job management flexibility.



Instructor
Notes

- | | |
|-----------------|---|
| Slide | Use slide 1-12 to repeat the Module Objectives. |
| Conclude | Have students discuss what they learned in this module and how they might apply it at their work sites. |

■ Manage Jobs

Module Summary



Instructor
Notes

Manage Events

Module Objectives

Slide 2-1



After this module, you will be able to:

- Send a Job On Hold Event
- Send a Job On Ice Event
- Send a Kill Job Event
- Send a Change Status Event
- Send a Force Start Job Event
- Send a Comment Event
- Send a Set Global Event

Module Overview

Every day at the close of business, after RBC has completed its daily transactions, you must run an end-of-day program set. Assume this program set contains a series of jobs named job1 through job20. About halfway through this set of programs, data is written to tape (job9). However, today your colleague notifies you that the tape drive is out of tape. job9 is part of the critical path and cannot be skipped.

In this Module, you will learn how the Event Processor drives Unicenter AutoSys JM scheduling. As your schedules evolve, events such as "start a job" or "kill a running job" are posted to the database. The Event Processor periodically checks the database and processes those events. At times like the one described in the previous paragraph, there may be a need to manually interfere with the scheduling process, which you accomplish using **Send Event**. There are many events that can be posted using Send Event. In this module, you will learn what an event is and how to use events by sending them to the database.



Instructor
Notes

Slide
Discuss
Overview
Problem

Use slide 2-1 to introduce Module Objectives

Use the scenario presented in the Overview to explain to students "what's in it for me" and why they should learn this module.

Slide 2-2



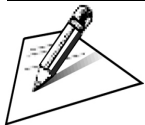
Task 1: Send a Job On Hold Event

Term	Definition
<i>Event</i>	An event is any type of status change or service request posted in the database and processed by the Event Processor.
<i>On Hold</i>	A job on hold will not run until a JOB_OFF_HOLD event or a FORCESTART_JOB event is sent for that job. The logic expressed by a job on hold still impacts jobs downstream of it.
<i>Send Event</i>	A command that permits you to manually post a change in a job's status to the database, to force the processing of that job or subsequent jobs.

Events are status changes or service requests posted in the database and processed by the Event Processor. Events could include, but are not limited to, the following:

- requests to kill a running job
- requests to force start a job
- changes to a global variable
- modifications to the job definition of an existing job

Sending events allows you to manually override system settings, a feature you may find handy for troubleshooting system hardware, or for testing job definitions before running them in a real environment.



Instructor
Notes

Slide Use slide 2-2 to introduce Task 1. Note there are no further slides for this task. Define terms.
Invite students to consider times when a manual override was required on their jobs.

■ Manage Events

Task 1: Send a Job On Hold Event

For example, before running RBC's nightly back-up to tape job, you may need to manually confirm the presence of tape media in the drive. You could send a `JOB_ON_HOLD` event using the Send Event command, giving you time to physically confirm the drive is ready before starting the job and producing an error, should the drive in fact, be off-line.

The Send Event command can be sent using the Send Event tool on the Scheduler Console in Windows, or on the Job Activity Console in UNIX.

You can select multiple jobs in the Scheduler Console before you open the Send Event tool. If you do so, the Send Event tool will send the chosen event for all of the selected jobs.

Using the Send Event tool, you can manually override job definition settings.

In addition, you have the option of submitting the override to occur immediately, or at some future date and time. The default setting is **Now**. Unless you select **Future**, all events sent using this dialog will be processed immediately.



Instructor
Notes

Scheduler Console in Windows



Instructor
Notes

Mention

Show the differences in the UNIX and Windows screens. Encourage the students to take notes in the space provided.

The Send Event tool default setting is Now. Students can submit Future events by selecting Future. This should be reinforced throughout the Module.

Job Activity Console in UNIX

Send Event

Event Type

- Start Job
- Force Start Job
- Change Status
- Job On Hold
- Job On Ice
- Change Priority
- Job Off Hold
- Job Off Ice
- Set Global
- Comment
- Kill Job
- Send Signal
- Stop Demon

☐ Cancel Previously Sent Event ☐ Match on Time

Job Name

Now Future Date 10/28/2003 (MM/DD/YYYY) Time 08:45 (hh:mm) A.M. P.M.

Comment

AUTOSERV Instance U45

Global Name Global Value

Signal Queue Priority

Status Running Send Priority Normal High

Execute Cancel



Instructor
Notes



Interactive Demonstration

Task Purpose: Place a job on hold in UNIX at some future date.

Note • Continue to follow the job naming convention specified at the start of this course by beginning each job with your name, followed by an underscore mark, a “u” to indicate UNIX, and then the job name: *name_[u|w]_jobname*.

- 1 From the **Job Activity Console**, select your *name_u_filewatcher1* job.
- 2 Click **Send Event**. The Send Event tool opens.
- 3 Select **Job On Hold**.
- 4 Click **Future**.
- 5 Set the desired future date and time. For the purposes of this example, select a date and time that is approximately five minutes from the time you are performing this task.
- 6 Click **Execute** and click **Yes** to continue.

The status column will update to ON_HOLD for the job you specified five minutes from now.

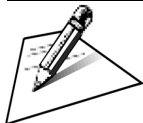


Interactive Demonstration

Task Purpose: Put a job on hold in Windows at some future date and time.

Note • Continue to follow the job naming convention, *name_[u|w]_jobname*.

- 1 From the **Scheduler Console**, select your *name_w_filewatcher1* job.
- 2 Click **Send Event**.



Instructor
Notes

■ Manage Events

Skill Practice

- 3 From the **Send Event** tool, select **Job On Hold**.
- 4 Select **Future**.
- 5 Set the desired time and date. For the purposes of this example, use a date and time that is five minutes from now.
- 6 Click **Send**.

Note • You will receive a confirmation message informing you that the request has been completed.

- 7 Close the Send Event tool.
- 8 After you have submitted the event, click **Refresh**.
- 9 Five minutes after you submit this event, you will see a status of ON_HOLD in the Status column for this job.



Skill Practice

Task Purpose: Take a job off hold. To take a job off hold, you follow nearly the same procedure as putting it on hold, except you select **Job Off Hold** from the Send Event tool, so there is no Interactive Demonstration for this task.

- 1 Start the job called **box1**.
- 2 For the job called **filewatcher1**, send a **Job Off Hold** event.
- 3 Examine the Console for the status of **filewatcher1**.



Perform the Skill Practice

Since removing a job from hold is so similar to placing it on hold, we skipped the interactive demonstration. Give students time to process the problem, coaching any who struggle.

Instructor
Notes

Task Summary

You now know what a Unicenter AutoSys JM event is and have practiced sending the first type of event: "on hold." Jobs placed on hold stop all automated processing scheduled to occur "downstream" of those jobs. There is a way to temporarily suspend a job so that processing of subsequent jobs may proceed. That event is called ON_ICE and will be covered in the next task.



Instructor
Notes

Close the Task by summarizing key concepts and checking understanding.

Slide 2-3



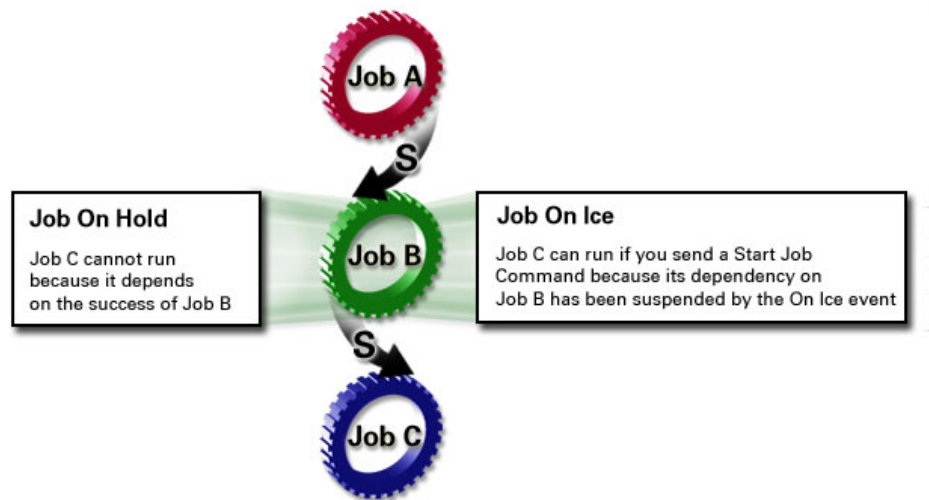
Task 2: Send a Job On Ice Event

Term	Definition
<i>On Ice</i>	The job will not run until a Job Off Ice Event is sent <i>and</i> all of its starting conditions occur again. The logic expressed by a job on ice is removed from the jobstream.

Slide 2-4



A job placed *on hold* and a job placed *on ice* seem similar at first, but have different impacts on jobs *downstream* of them. The distinction between the two terms is based on how Unicenter AutoSys JM evaluates the job's starting conditions. When taken *off hold*, the job will run as long as all of its starting conditions were already met. But when taken *off ice*, the job will not run until all of its starting conditions occur again, even if all of its starting conditions were previously met.



Instructor
Notes

Slide Use slide 2-3 to introduce Task 2.
Define terms.

Slide Use slide 2-4 to compare On Hold and On Ice: The gears represent a jobstream, with Job C depending on the success of Job B, which in turn depends on the success of Job A to start. If you place Job B on hold, Job C also is affected since it is dependent on a SUCCESS status. However, if you place Job B on ice, you can start Job C because an on ice status also removes the dependencies from the job stream.

In other words, the logic expressed by a job *on hold* still impacts any of its dependent jobs, while the logic expressed by a job *on ice* is removed from the jobstream. As the previous illustration shows, when a job is placed on hold, any jobs dependent on it cannot run, since the dependent condition has not been met. But, a job dependent on the job on ice can be started using a Start Job event, because the logic expressed by the dependency is halted until the job is removed from ice.



Interactive Demonstration

Task Purpose: Place a job on ice in UNIX.

- 1 From the **Job Activity Console**, select your **filewatcher1** job.
- 2 Click **Send Event**. The Send Event tool opens.
- 3 From the **Event Type** group, select **Job on Ice**.
- 4 Click **Execute**.
- 5 You will receive a confirmation message informing you that the request has been completed. Click **Yes** to clear the message.
- 6 Close the Send Event tool.
- 7 Examine the Job Activity Console. You will now see a status of ON_ICE in the Status column for this job.



Interactive Demonstration

Task Purpose: Place a job on ice in Windows.

- 1 From the **Scheduler Console**, select your **filewatcher1** job.
- 2 Click **Send Event**.



Instructor
Notes

- 3 From the Send Event tool, select **Job on Ice** and click **Send**.

Note • You will receive a confirmation message informing you that the request has been completed.

- 4 Close the Send Event tool.
- 5 After you have submitted the job, click **Refresh**.
- 6 You will now see a status of ON_ICE in the Status column for this job.



Skill Practice

Task Purpose: Take a job off ice. To take a job off ice, you follow nearly the same procedure as putting it on ice, except you select **Job Off Ice**.

- 1 Start your **box1** job.
- 2 Note the current status for **box1** and **command1**.
- 3 For the job called **filewatcher1**, send a **Job Off Ice** event.
- 4 Note the new status for **filewatcher1**.

Task Summary

So far, you have learned how to send two events: ON_HOLD and ON_ICE, as well as how the two events impact job processing. Both of these events have this in common: they prevent a job from starting. In the next task, you will learn how to stop a job that is already running.



Instructor
Notes

Perform the Skill Practice

Before starting the Skill Practice, ask students what status they think will be displayed when the job is taken off ice?

Have them perform the Skill Practice, and coach anyone who struggles.

Discuss results. Who guessed the right status and who did not? Discuss incorrect statuses.

Close Additional Review

the Task and check students' understanding.

If time permits, pose this question to your class:

When a job is placed on ice, the job dependent on it to start will immediately run one time only, which could be a problem. How can you work around it?

Slide 2-5



Task 3: Send a Kill Job Event

Term	Definition
<i>Kill job</i>	A request (or GUI option) that sends a KILLJOB event for a running job.

The KILLJOB event is used to kill jobs that are RUNNING. Sometimes, jobs may run too long or require too many system resources. Sending a KILLJOB event will kill the job to free up the system.



Skill Practice

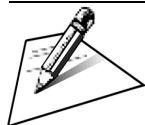
Task Purpose: Kill a job in UNIX. This exercise is a combined Skill Practice and Interactive Demonstration. You already know how to create basic command jobs; practice that skill by creating a command job that you will then kill as part of the demonstration.

Note • In this exercise, you must first create the job called *name_u_sleep1*. Then, start the *sleep1* job. Make sure the status is RUNNING. When the job runs, watch the status indicator in the Event Report section of the Job Activity Console. (Remember: *name_* represents your last name.)

- 1 Create a new command job called *name_u_sleep1*.
- 2 Set the job to execute the following command:

```
sleep 90
```

This command instructs UNIX to wait 90 seconds before completing, so you have 90 seconds to send the KILLJOB event to complete this demonstration.



Instructor
Notes

Slide Use slide 2-5 to introduce Task 3. Note, there are no further slides for this task.

Define terms.

Ask students why they might ever need to kill a running job? Use their responses as you present the remainder of the Task to add realism.

- 3 Run your **sleep1** job. When the job is running, you should see the status in the Event Report section update to **RUNNING**.
- 4 Click **Send Event**. Confirm **Now** is selected.
- 5 From the Send Event tool, select **Kill Job** and click **Execute**.
- 6 You will receive a message. Click **Yes** to clear the message.
- 7 Close the Send Event tool.

You will now see a status of **TERMINATED** in the status column for this job.

Skill Practice



Task Purpose: Kill a job in Windows. This exercise is a combined Skill Practice and Interactive Demonstration. You already know how to create basic command jobs; practice that skill by creating the command job to run the Windows calculator, which you will then kill as part of the demonstration.

- 1 Create a new command job called: *name_w_calc3*
- 2 Set your job to execute this command: `calc.exe`
- 3 Start your **calc3** job and watch the Status column for this job update to **RUNNING**.
- 4 Click **Send Event**.
- 5 From the Send Event tool, select **Kill Job** and click **Send**.
- 6 You will receive a message. Click **OK** to clear it.
- 7 Close the Send Event tool.



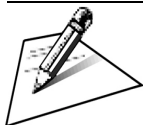
Instructor
Notes

- 8 After you have submitted the event, click **Refresh**.
- 9 You will now see a status of **TERMINATED** in the Status column for this job.

Note • The Windows Calculator has now closed.

Task Summary

You now know why jobs might need to be killed and how to send the KILLJOB event to the database. So far, you have learned three types of events, discussed how they should be used, and practiced sending them to the database. All of the events covered so far imply “real” conditions. The next event you will learn covers situations in which reality differs from what is being reported by the database.



Instructor
Notes

Close the Task by summarizing key concepts and checking understanding.

Slide 2-6



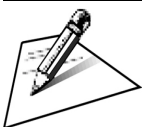
Task 4: Send a Change Status Event

Term	Definition
<i>Change Status</i>	Forces a change in status to the job specified. Used with the Status list (bottom left corner of the Send Event tool) from which you specify the target status for the job selected.

The `CHANGE_STATUS` event forces a change in the *posted* status of the selected job. This distinction is important: the *posted* status may differ from the job's *real* status. Because Unicenter AutoSys JM manages job status changes internally, this option should ordinarily not be used. However, it is useful for verifying job logic or for recovering from error situations.

Example: Consider a job that prints paychecks. After a sufficient period of time has elapsed, you check the job's status and are surprised to learn that Unicenter AutoSys JM shows the job as still `RUNNING`. A physical check of the printer shows that the job completed successfully; you have the exact number of paychecks you expected. You need to change the posted status of this job to match its real status, to continue processing subsequent jobs. You can do this by sending a `CHANGE_STATUS` event.

WARNING • Before you issue a `CHANGE_STATUS` event, you should always run the `chase -A` command, to verify that the jobs the database indicates are `RUNNING`, are indeed running. The `chase` command is covered in more detail in *Module 4, Use Basic Commands*.



Instructor
Notes

Slide	Use slide 2-6 to introduce Task 4. Define terms. Reinforce that changing a job's status in this way is for reporting purposes only; it may not be what is actually happening. Consider reasons why the reported status may differ from actual status.
Warn	students that the paycheck scenario is example only; Module 4 covers in more detail what should actually be done before changing a status, just to be sure. Note: This was the sequence agreed upon by SMEs during Development; if class use makes the case for moving the <code>chase</code> command to this module, please contact Curriculum Development.
Discuss	This would be a good time to mention "priority." A High Send Priority makes commands register right away, in other words, without delay.

Using the CHANGE_STATUS event requires you to select an appropriate status from the Status List on the Send Event tool. There are two possibilities:

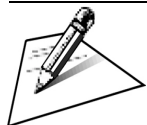
- When changing a job's status for the first time, use the Status List to select the desired status to be set.
- When returning a job's status to its former setting, use the Status List to select the original status prior to issuing the CHANGE_STATUS event.

Slide 2-7



Discuss

Given the emphasis on the distinction between *posted* and *real* status, do you think sending a CHANGE_STATUS event of TERMINATED is identical to sending a KILLJOB event?



Discussion

Pose the question provided. Ask students to offer their guesses before showing them the right answer (perform a brief Instructor only demo and point out that the status is still RUNNING).

Instructor
Notes



Interactive Demonstration

Task Purpose: Change the status of a job in UNIX.

- 1 From the **Job Activity Console**, select your **commandfail** job and click **Send Event**.
- 2 Select **Change Status**.
- 3 From the **Status** list, select **Success** and click **Execute**.



Interactive Demonstration

Task Purpose: Change the status of a job in Windows.

- 1 From the **Scheduler Console**, select your **commandfail** job and click **Send Event**.
- 2 Select **Change Status**.
- 3 From the **Status** list, select **Success** and click **Send**.

Note • You will receive a confirmation message informing you that the request has been completed.

Task Summary

You now know how, as well as why you might need to change the status of a job. In the next Task, you will learn how to send a FORCE START JOB event, which differs from the previously-covered START_JOB event as follows: START_JOB will not start jobs whose starting conditions have not been met; FORCESTART_JOB will.



Close the Task by summarizing key concepts and checking understanding.

Instructor
Notes

Slide 2-8



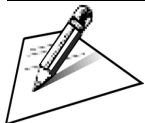
Task 5: Send a Force Start Job Event

Term	Definition
<i>Force Start Job</i>	An event that starts a job immediately, even if its starting conditions have not yet been met.

While the START JOB command starts only a job whose starting conditions have already been met, the FORCE START JOB command starts any job, even if its starting conditions have not been met.

Example

In Module 1, you created a basic box job called box1. In it, you placed your basic command and filewatcher jobs. As you have already learned, your command and file watcher jobs cannot run unless the box containing them is already running. Suppose you wish to test the logic you set for a job that happens to be in this box, which is not running. Clicking **Start Job** will do nothing, since the box is not running. Therefore, you use the **Force Start Job** command to essentially ignore all of the other jobs and starting conditions, and start only the selected job.



Instructor
Notes

Slide	use the slide to introduce Task 5. Define terms.
Contrast	<p>To describe the distinction between Start Job and Force Start Job, use Arun Prabhu's metaphor of a polite request and a demand. The Start Job event is polite, but the Force Start Job is rude.</p> <p>Start Job is like asking "May I please start this job, if everything is OK, such as the date and time, the dependent conditions, and so on?" If everything is OK, the job will start.</p> <p>Force Start Job demands that the job be started right now - you do not care what conditions have not been met, you just want it done.</p>



Skill Practice

Task Purpose: Practice force starting a job in UNIX or Windows. In a previous demonstration, you created a box job called **box1** and to it, added the jobs called **command1** and **filewatcher1**.

- 1 Ensure that the **command1** job is still inside the **box1** job. If it is not, add it.
- 2 Use the **Start Job** command to start the **command1** job and note its status.
- 3 Use the **Force Start Job** command to start the **command1** job and note its status.
- 4 Discuss the results of steps 2 and 3.

Task Summary

In this task, you learned the distinction between starting and force starting jobs, as well as how to send a FORCE START JOB command. In the next Task, you will learn how to send comments, which annotate specific job runs, not the job definitions.



Close the Task by summarizing key concepts and checking understanding.

Instructor
Notes

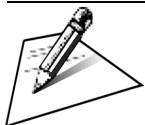
Slide 2-9



Task 6: Send a Comment Event

Term	Definition
<i>Comment</i>	Associate a comment with a specific job run. Comments are sent as events and are stored in the database, permitting you to view the last comment sent, using the Event Report window on the UNIX Job Activity Console. Note • Do not confuse this with the Description field on the UNIX Job Definition or the Windows Job Editor tools.

Sending a comment is similar to sending any other event, except you select the **Comment** option from the Send Event tool. It is useful for documenting job run anomalies; perhaps a job had to be manually started due to some hardware malfunction, or perhaps a job run only on occasion was requested by a Senior Manager.

Instructor
Notes

Slide	Show the slide to introduce the Task. Define terms.
Reinforce	Sending a comment is NOT identical to completing the Description field on the Job Editor or Job Definition screens. Comments are stored in the database per job run. Most companies have specific policies for documenting job execution. Comments help you enforce those policies. Invite the class to give their own examples for Comments.



Skill Practice

Task Purpose: Practice sending a comment in UNIX or Windows.

- 1 Use the Send Event tool to send the following comment for your **command1** job:

Future event for command1
- 2 Send the event.

Note • You will receive a confirmation message informing you that the request has been completed.

Task Summary

Now you know how to attach a comment to a job run. In the next Task, you will learn how to automate jobs based on “what if” conditions, such as Manager Approval or an error condition.



Instructor
Notes

Close the Task by summarizing key concepts and checking understanding.

Slide 2-10

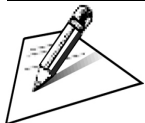


Task 7: Send a Set Global Event

Term	Definition
<i>global variable</i>	Global variables can be used to create job dependencies. The value of the expression must evaluate to TRUE for the job dependency to be satisfied.

Examples

- If you have a set of jobs in a box that is supposed to run only on special occasions, such as only on your manager's approval, you would set the value of a global variable you named "manager-ok" to "OK," and make the top-level box job dependent on this global variable, to start the jobstream.
- Global variables can be handy for backing out of error conditions. To return a set of mission-critical files to the last backed up version, you could run a job called `job_restore` that runs only when `job_writedata` must be reversed. You could use a global variable called `Correct` and set it to **yes** to start the jobstream that corrects the error situation.



Instructor
Notes

Slide Show the slide to introduce the Task.
Define terms, review the Examples, invite the class to offer their own.



Interactive Demonstration

Task Purpose: Set a global variable in UNIX.

- 1 From the **Job Activity Console**, click **Send Event**.
- 2 From the Send Event tool, select an Event Type of **Set Global**. The Global Name attribute is enabled.
- 3 Type a name for the global variable, no spaces:
newyork
- 4 Type a value for this variable in the field labeled **Global Value**:
bigapple
- 5 Click **Execute**. Click **Yes** to confirm and clear the message.
- 6 To verify that the value associated to this global variable was really saved, use the `autorep` command at the command prompt:
autorep -G newyork

The screen displays the name of the global variable, its current value, and the date and time this value was set.

The `-G` option specifies a global variable named “newyork” but you could use the `ALL` value to report on all global variables.



Instructor
Notes

Ask students how they might use global variables at their work sites.



Interactive Demonstration

Task Purpose: Set a global variable in Windows.

- 1 From the **Scheduler Console**, click **Send Event**.
- 2 From the Send Event tool, select **Set Global**.
- 3 In the **Global Name** field, type:
`global1`
- 4 In the **Global Value** field, type the following and click **Send**:
`value1`

Note • You will receive a confirmation message informing you that the request has been completed.

- 5 Close the Send Event tool.
- 6 After you have submitted a job, click **Refresh**.

You will now see a status of SUCCESS in the Status column for the job you just ran.

Verify

- 1 Open the AutoSys Command Prompt.
- 2 Type `autorep -G global1` and press ENTER.

Notice the global variable called `global1` was created with the appropriate value, `value1`.



Instructor
Notes

Skill Builder: Class Discussion



Business Problem

It is 5 PM on a weekday. Every day at the close of business, after RBC has completed its daily transactions, you must run an end-of-day program set. Assume this program set contains a series of jobs named job1 through job20. About halfway through this set of programs, data is written to tape (job9). However, today, your colleague informs you that the tape drive is out of tape. job9 is part of the critical path and cannot be skipped. What should you do?

After job9 completes, job10 prints a summary of the day's activities. The printer paper is jammed. Since job10 is not part of the critical path, it may be skipped. What should you do?

After the printer jam is cleared and job10 completes successfully, job11 is supposed to start. However, perhaps due to the paper jam or other network problem, job10 is showing an incorrect status, preventing job11 from starting. What should you do?



Instructor
Notes

Perform the Skill Builder

Make sure students understand the problem and its directions.

This is not a hands-on activity, but a group discussion.

Instructor discretion: consider dividing the class into small groups and ask each group to present a solution, or discuss the problem as a class.

Solution

Job 9 should be placed on hold.

Job 10 should be placed on ice.

Job 10 should receive a change status to SUCCESS, or Job 11 could be force started.

Slide 2-11

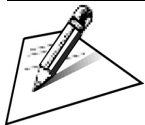


Module Summary

You should now be able to:

- Send a Job On Hold Event
- Send a Job On Ice Event
- Send a Kill Job Event
- Send a Change Status Event
- Send a Force Start Job Event
- Send a Comment Event
- Send a Set Global Event

In this Module, you learned how to send events to manually override the information stored in the Unicenter AutoSys JM database and when each type of event could be used. You practiced sending events in UNIX and Windows, and discussed scenarios for applying them. In the next Module, you will learn how to schedule jobs.



Instructor
Notes

Slide Show the slide to reinforce the Module Objectives.
Review all key terms and concepts. Check for understanding.

■ Manage Events

Module Summary



Instructor
Notes

Use the Scheduler or Job Activity Console

Module Objectives

Slide 3-1



After this module, you will be able to:

- Create Filters
- Sort Columns in the Console

Module Overview

The Windows *Scheduler Console* — or the UNIX *Job Activity Console* — allows you to monitor jobs from multiple instances in real time. The term Console will be used throughout this Module to refer to either version; where a specific operating system is referenced, the proper term will be used.

The Scheduler Console lets you view any jobs that are defined in the database, whether or not they are currently active. The Job Activity Console offers similar options; however, filters cannot be saved.

You can create *filters* that allow you to control the jobs displayed in the Console. These filters are based on various parameters, such as the current job state, the job name, and the machine on which the job runs. You can change the current filter to change the jobs displayed in the console.

In addition, from the Console, you can select any job and view more detailed information about it, including its starting conditions, dependent jobs, and autorep reports.



Slide Show slide 1 to introduce the Module Objectives.

Instructor
Notes

Slide 3-2



Task 1: Create Filters

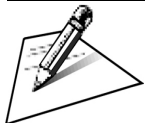
Term	Definition
<i>Filters</i>	Provide ways for you to control which jobs are displayed in the Console.
<i>Default Filter</i>	Includes all jobs, with all statuses, on all defined machines, and for all locally-defined (installed) instances.
<i>Filter Editor</i>	A tool used to define, modify, and delete filter conditions. (Is available only in Windows.)

Examples

Controlling the jobs the Console displays at a specific point in time can be useful in situations where you must monitor multiple high-level jobs simultaneously, such as special requests from senior management, which you could do by creating a job name filter.

Alternatively, you could —in response to an equipment breakdown — create a filter based on a FAILURE status, to view the jobs that were affected by the unavailability of that particular system resource.

Since Windows permits you to save defined filters, you can define filters for a variety of situations, then view the Scheduler Console by selecting the desired filter from a list.



Instructor
Notes

- Slide** Show slide 2 to introduce the Task and define terms. Note there are no additional slides for this task.
- Discuss** the examples. Discuss students' personal examples, if any are offered.

■ Use the Scheduler or Job Activity Console

Examples

The **Filters** menu on the Scheduler Console opens the Filter Editor. In UNIX, the **View ► Select Jobs** menu on the Job Activity Console opens the Job Selection tool. The following table describes filter parameters:

Parameter		Description
Names	All Jobs	Selects all jobs that meet the filter criteria.
	Jobs Named	Selects only the jobs with the name or names you type in the text box to the right of this option. Note • As you enter each name, press ENTER.
	Jobs in Box Named	Also selects the jobs with the name or names you type in the text box to the right of the button, AND enables the Maximum Depth field: All: The box specified and all of its direct descendants (default). 0: Only the top-level box specified. 1-20: The box specified and the selected number of levels of nesting.
Status	All Statuses	You can select the jobs you wish to display based on their current status or on specific statuses, such as ON HOLD. You may select any combination of statuses.



Instructor
Notes

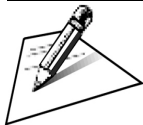
Parameter	Description
Machine/Instance	All Machines You can select jobs to display based on the machine on which they ran.
	All Instances You can select jobs to display based on the instance or instances to which the Console can connect.



Interactive Demonstration

Task Purpose: Create a filter for a box in UNIX.

- 1 From the Job Activity Console, choose **View ► Select Jobs**. The Job Selection tool is displayed.
- 2 Under the **Select by Name** group, select **Box Name**. This makes the text field available.
- 3 In the **Box Name** field, type:
`name_u_box1`
- 4 Click **Apply**.
- 5 Click **OK**. The Job Selection tool closes. The Job Activity Console now shows the jobs that meet the selection criteria you set.



Perform the Demonstration

At Step 3, explain what other ways students can filter jobs, other than directly typing a job name.

Instructor
Notes



Interactive Demonstration

Task Purpose: Create a filter for a box in Windows.

- 1 From the Scheduler Console, choose **Filters ▶ Filter Editor**. The Filter Editor opens.
- 2 On the **Names** tab, select **Jobs in Boxes Named**.
- 3 Type the following in the **Box name** field:
`name_w_box1`
- 4 Choose **File ▶ Save** and name the filter: `name_w_filterbox`
- 5 Click **OK**.
- 6 Exit the Filter Editor. The filter you just created is now listed in the **Filter** list.
- 7 From the **Filter** list (upper-left corner), select **filterbox**. Notice that all jobs within **box1** are now displayed.



Interactive Demonstration

Task Purpose: Create a filter for an event in UNIX.

- 1 From the Job Activity Console, choose **View ▶ Select Jobs**. The Job Selection tool is displayed.
- 2 From the **Select by Status** group on the Job Selection tool, ensure that only **Success** has been selected.
- 3 Click **Apply**.
- 4 Click **OK**. The Job Activity Console display is updated to show only jobs with a **SUCCESS** status.



Instructor
Notes



Interactive Demonstration

Task Purpose: Create a filter for an event in Windows.

- 1 From the Scheduler Console, choose **Filters ▶ Filter Editor**.
- 2 Click the **Status** tab. Ensure that only **Success** has been selected.
- 3 Choose **File ▶ Save** and name the filter: `name_w_filterevent`
- 4 Click **OK**.
- 5 Exit the **Filter Editor**.
- 6 From the **Filter** list (upper-left corner), select **filterevent**. Notice that all jobs with a status of **SUCCESS** are now displayed.



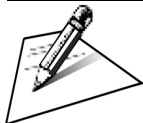
Skill Practice

Task Purpose: Practice creating filters using UNIX or Windows. Assume that a major piece of networking equipment broke down today and that you wish to view all jobs with a **FAILURE** status, to check their parameters.

Show all jobs with a status of **FAILURE**.

Task Summary

You now know how to use the Windows Scheduler Console, or the UNIX Job Activity Console, to filter the jobs to display. In the next Task, you will learn how the Console can be customized further.



Instructor
Notes

Close the task by reviewing concepts and checking for understanding.

Slide 3-3



Task 2: Sort Columns in the Console

In the Console, jobs are displayed in the order for which they are returned from the database. You can modify the sort order by choosing a primary sort criterion.

In UNIX, the Job Activity Console provides the following options:

Sort Criteria	Description
Job Name	Sorts jobs by name in ascending order (alphanumeric).
Job Status	Sorts jobs by their current status in ascending order.
Machine Name	Sorts jobs by the machine on which they will run or have run, in ascending order.
Start Time	Sorts jobs by the starting time for the most recent job execution or the time it is scheduled to run.
End Time	Sorts jobs by the ending time for the most recent job execution or the time it is scheduled to end.

Note • Use the default sort order (which is no sorting) if you wish to view levels of box jobs. If you sort jobs, you will have no way to recognize which boxes contain which jobs, because you will have removed the indentation that shows job nesting.



Slide Show slide to introduce Task. There are no additional slides for this task.

Instructor
Notes

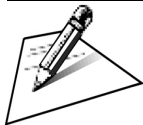
In Windows, you can sort on the fly using any column. The Scheduler Console provides the following options:

Sort Criteria	Description
Job Name	Sorts jobs by name in ascending order (alphanumeric).
Instance	Sorts each job by its instance name in ascending order.
Status	Sorts each job by its current status in ascending order.
Last Change	Sorts jobs by the date of their last change.
Machine	Sorts jobs by the machine on which they will run or have run, in ascending order.
Type	Sorts jobs by type in ascending order.
Start Time	Sorts jobs by the starting time for the most recent job execution or the time it is scheduled to run.
Job ID	Sorts jobs by the order in which they were created.

Interactive Demonstration

Task Purpose: Sort columns in the Job Activity Console.

- 1 From the Job Activity Console, choose **View ► Select Jobs**. The Job Selection tool is displayed.
- 2 From the **Sort Order** group, select **Job Name**.
- 3 Click **Apply** and then **OK**. The Job Activity Console now shows jobs sorted by job name.



Instructor
Notes



Demonstration

Task Purpose: Sort columns in the Scheduler Console. Your instructor will demonstrate that by clicking on the column titles, you can change your sort order.



Task Summary

In this Task, you learned how to customize the Summary Area Layout according to a primary sort criterion. In addition to filters, sorting the Console view can be an effective tool for managing your jobs.



Close the task by reviewing concepts and checking for understanding.

Instructor
Notes



Skill Builder: Schedule RBC's Accounting Jobstream

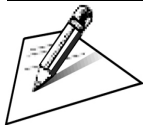
Business Problem

RBC's internal auditors have asked you to create and run special jobs to verify accounting records. Since these jobs are not part of your usual work load, you wish to watch them yourself. Initially, you would like to view these jobs based on their status. Then, for jobs that show a status of FAILURE, you wish to view them by their end times.

Hint

Apply what you have learned in this course so far to solve this problem. You must first create the jobs to run. Use the following job names in your solution:

Job Name	Description
name_acctbox03	contains the jobs called name_acct04_03 and name_accountfile
name_acct04_03	runs account.bat
name_accountfile	watches for account.txt
name_acctbox02	contains the jobs called name_finfile and name_accounting02
name_finfile	watches for financial.txt
name_accounting02	runs acct_annual.bat



Instructor
Notes

Solution students should create and run the jobs listed in the table. Next, they should filter the jobs by the two box job names to view just these jobs. Finally, they should sort the failed jobs by their end times.

Ask students what they would do differently if they knew these accounting jobs were not in a box?

Slide 3-4



Module Summary

You should now be able to:

- Create Filters
- Sort Columns in the Console

In this module, you learned to use the Scheduler and Job Activity Consoles. These consoles are used to create filters allowing you to control the jobs displayed. In the next module, you will learn to use basic commands in both UNIX and Windows.



Instructor
Notes

Close the Module by reviewing the Module Objectives.

Use Basic Commands

Module Objectives

Slide 4-1



After this module, you will be able to:

- View a Job Detail Report for Multiple Jobs
- View Chase Output for All Running Jobs
- Verify that the Event Processor and Event Server are Running
- View Current Version of Unicenter AutoSys JM
- Report Job Dependencies
- View the Event Processor Log or Remote Agent Log
- Send Events

Module Overview

Although this course focuses primarily on using the graphical user interfaces in UNIX and Windows, there are some basic commands any Operator might find useful. In this module, you will learn to start the command line interface and send some basic commands.



Instructor
Notes

Slide Show slide 1 to introduce the Module Objectives.

Explain why knowledge of basic commands is provided in a course that focuses primarily on GUI usage. (It is much faster.)

Refer students to the Learning Path; specifically, to UA200, if they express interest in learning more commands.

Slide 4-2



Task 1: View a Job Detail Report for Multiple Jobs

In *Module 1, Manage Jobs*, you performed an exercise that included using the Job Detail Report. Using the Windows or UNIX GUI, you can use the Job Detail Report tool to display a real-time report for only one currently selected job at a time.

You can display the Job Detail Report in two ways:

- **Summary:** displays a single line that shows the last or current execution of the job.
- **Event:** displays a detailed report that shows all the events and statuses from the last or current execution of the selected job.

However, to view job details for *more than one* job, you must use the `autorep` command at the command line interface.

The `autorep` command reports information about a job, jobs within boxes, machines, machine status, job overrides, and global variables.

Slide 4-3

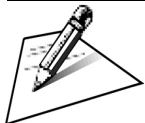


Syntax

```
autorep {-J job_name | -M machine_name | -G global_name}
```

```
[-s | -d | -q | -o over_num | -u] [-r run_num] [-L print_level]
```

```
[-t] [-D data_server:database | -D TNSname]
```



Instructor
Notes

Slide Show slide 2 to introduce the Task.

Slide Show slide 3 to discuss command syntax. The following page features a table that explains each switch and option in more detail.

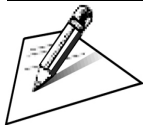
Command Parameters

Parameter	Description
-J job_name	<p>Indicates that a Job Report is desired.</p> <p>job_name specifies the name of the job on which to report. Any valid job name may be specified.</p> <p>To report on all jobs, specify ALL. The percent (%) character may be used in the job name as a wildcard. (For example: %box% will select all jobs containing the string "box")</p>
-M machine_name	<p>Indicates that a Machine Report is desired. The report lists the machine's Max Load, Current Load, and Factor.</p> <p>machine_name specifies the machine on which to report. This may be a virtual machine, a real machine, or ALL for all machines; the machine must be defined.</p>
-G global_name	<p>Indicates that a global variable report is desired. The report lists the variable name, value, and last modification date.</p> <p>global_name specifies the name of a global variable that has been set using the sendevent command or the Send Event dialog. In the specification, you can use ALL or wildcard characters.</p>
-s	Indicates that a Summary Report is desired. This is the default.
-d	Indicates a Detail Report is desired.



Instructor
Notes

Parameter	Description
-q	Indicates a Query Report is desired, providing the current job or machine definition, in JIL format, as it exists in the database.
-o over_num	Indicates an Override Report is desired, providing the overrides for the specified override number (over_num) and job_name. If the most current override is desired, the value of over_num should be zero. The job attributes and their associated overrides are displayed to standard output. You must supply a job name (-J job_name) with this option.
-r run_num	Indicates a report is desired for a specific job run (run_num). This option can only be used with the -s and -d options. If this option is omitted, or run_num is zero, the most current job run is reported. A minus sign (-) can be used before the run_num value to indicate a relative counter for a past job run, relative to the current run number. For example, the option -r -2 would generate a report for the job run two runs back.
-t	Requests that the time zone, if one is specified in the job definition, appear in the report. If requested, the time zone will appear in parentheses beneath the job name and the Time column of the report will be based on the specified time zone.
-u	Requests an output of Create/Modified User Data. Time fields of a job are displayed.



Instructor
Notes

■ Use Basic Commands

Command Parameters

Parameter	Description
-L	(Job reports only.) Indicates the number of levels of nesting for boxes for which the specified information should be listed. For example, a level of 2 means list the information for the specified job (a box) as well as two levels of jobs within that box.
-D data_server: database	<p>Indicates the name of the Sybase® data server, and the specific database within it, to be searched for the specified information. Normally, <code>autorep</code> consults the configuration file (<code>\$AUTOUSER/config.\$AUTOSERV</code>) to determine which database for which it is to connect. This option enables <code>autorep</code> to report on any Event Server on the network.</p> <p>Indicates the name of the Sybase data server, and the specific database within it, to be searched for the specified information. Normally, <code>autorep</code> consults the configuration file (<code>\$AUTOUSER/config.\$AUTOSERV</code>) to determine which database for which it is to connect. This option enables <code>autorep</code> to report on any Event Server on the network.</p>
-D TNSname	Indicates the TNS alias name of the Oracle® data server to be searched for the specified information. Normally, <code>autorep</code> consults the configuration file (<code>\$AUTOUSER/config.\$AUTOSERV</code>) on UNIX, or the Windows NT Registry, to determine which database for which it is to connect. This option enables <code>autorep</code> to report on any Event Server on the network.



Instructor
Notes

Example

Job Name	Last Start		Last End		ST	Run	Pri/Xit
arun_job1	04/28/2003	20:05:22	04/28/2003	20:05:23	SU	105/1	
pat_unix_box1	05/07/2003	14:45:43	05/07/2003	15:39:06	SU	128/0	
pat_unix_command1	05/07/2003	15:39:04	05/07/2003	15:39:05	SU	128/2	
pat_unix_filewatcher1	05/07/2003	14:45:45	05/07/2003	14:46:46	IN	128/1	
pat_commandfail	05/07/2003	14:30:02	05/07/2003	14:30:04	FA	126/1	1
sales_report_box	05/06/2003	11:29:01	05/07/2003	14:27:04	FA	117/1	1
sales_report_command	05/06/2003	11:29:03	05/06/2003	11:29:04	FA	117/1	1
sales_report_filewatcher	05/06/2003	11:29:03	05/07/2003	14:27:03	TE	117/1	250
pat_command1	05/07/2003	10:43:04	05/07/2003	10:43:05	FA	118/1	1
patty_command1	05/07/2003	14:45:23	05/07/2003	14:46:54	SU	127/1	
patty_filewatcher2	-----	-----	-----	-----	IN	0/0	
testreport	05/08/2003	13:49:17	05/08/2003	13:50:48	SU	129/1	

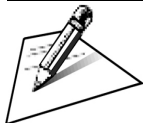
The command `autorep -J ALL` produces a report such as that shown in the graphic.

Start the Command Line Interface

In Windows, click **Start ▶ Programs ▶ Computer**

Associates ▶ Unicenter ▶ Unicenter ▶ AutoSys JM (*instance name*) ▶ Command Prompt.

In UNIX, go to the \$ prompt.



Instructor
Notes

Offer additional examples of the command, if time permits, as a Demonstration, rather than an Interactive Demonstration.



Interactive Demonstration

Task Purpose: Use the `autorep` command to show job details.

- 1 Open a command prompt.
- 2 Type the following command:

`autorep -J ALL`
- 3 Discuss the results.

Task Summary

You now know how to use the `autorep` command to show job details.



Instructor
Notes

Close the task by reviewing concepts and checking understanding.

Slide 4-4



Task 2: View Chase Output for All Running Jobs

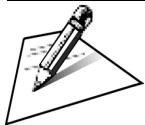
In a real business environment, you may frequently find that you need to verify that the jobs the database indicates are running, are actually running. Back in *Module 2, Manage Events*, we used a paycheck job to illustrate changing a job's status. In that example, Unicenter AutoSys JM showed the job as RUNNING, even though the printer produced the proper number of paychecks. Before issuing a change status event, you need to confirm that the job really is no longer running.

To do this, you use the `chase` command. This process also checks the associated Remote Agents.

The `chase` command checks the database for jobs in the STARTING or RUNNING state, and on which machine. The `chase` command passes a list of jobs that are supposed to be running on that client machine to a Remote Agent and instructs the Remote Agent to verify that the processes are indeed running.

For command jobs running on a UNIX machine, the Remote Agent also checks for the `pid` (process ID) of the UNIX process. The `chase` command also verifies that the Remote Agent is running. When the Event Processor is started, `chase` is automatically invoked. Since the `chase` command uses the same mechanism as the Event Processor to communicate with the Remote Agent machines, it gives an accurate picture of the system's state of health. When verifying that the Remote Agent is running, the `chase` command checks that the Remote Agent has a lock on the Remote Agent log file. (This is more reliable than checking the Remote Agent's process ID.)

Note • If you have disabled file locking on the client machine, `chase` will not be able to verify if a Remote Agent is running. Therefore, ensure that the directory specified by the `AutoRemoteDir` parameter in the configuration file is on a file system that has file locking enabled.



Instructor
Notes

Slide Show slide 4 to introduce the Task.

Since this is an introductory class, you may need to explain what "pid" is to UNIX novices.

Syntax

Slide 4-5



Syntax

`chase [-A] [-E]`

Parameters

- A option sends an alarm to alert the user that problems were found.
- E option will force a FAILURE of any job that is supposed to be running but is not.

The -E option will trigger an automatic restart of the job if the `n_retrys` attribute has been defined. The Event Processor must be running for `chase` to automatically restart jobs. Errors detected by `chase` are sent to standard output.

Running the chase process.

There are no jobs in STARTING or RUNNING state!

Chase is successful.

In some business cases, jobs may run longer than expected. If you suspect a job may be stuck, but still shows a status of RUNNING, you could issue the `chase -A` command to verify it. If you want to automatically restart the job if it is stuck, you would issue the same command with the -E option.



Slide Show slide 5 to discuss command syntax.

Instructor
Notes



Skill Practice

Task Purpose: Confirm that a job is no longer running.

- 1 Run chase.
- 2 Discuss the output.

Task Summary

You now know how to run the chase command to confirm jobs are actually running and not stuck in a processing loop.



Close the task by summarizing concepts and checking understanding.

Instructor
Notes

Slide 4-6



Task 3: Verify that the Event Processor and Event Server are Running

The `chk_auto_up` command determines if the Event Server (database) and the Event Processor are running. This facility is essential for locating the cause of problems, such as jobs not being started at the scheduled time. The Event Server and the Event Processor must both be running for jobs to start.

Slide 4-7



Syntax

`chk_auto_up [-Q]`

where:

-Q indicates that the command should output just the exit code without any descriptive message. This makes the command useful for inclusion in shell scripts. In this case, the return code is sufficient to indicate the status.

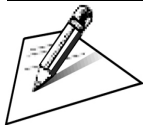
Return Code	Optional Message
0	Event Processor not running; No Event Server.
1	Event Processor not running; Event Server up.
2	Event Processor not running; Primary and Dual-Event Servers up.
10	Event Processor up; Event Server name invalid, probably because the Event Server (<code>EventServer</code> parameter in the configuration file) was correct when the Event Processor was started, but was corrupted before you ran: <code>chk_auto_up</code> .



- Slide** Show slide 6 to introduce the task.
- Slide** Show slide 7 to discuss command syntax.

Instructor
Notes

Return Code	Optional Message
11	Event Processor up; Event Server up.
12	Event Processor up; Primary and Dual-Event Servers up.
20	Shadow Event Processor up; Event Server name invalid (see return code 10).
21	Shadow Event Processor up; Primary Event Processor not running; Event Server up.
22	Shadow Event Processor up; Primary Event Processor not running; Primary and Dual-Event Servers up.
30	Primary and Shadow Event Processors up; Event Server name invalid (see return code 10).
31	Primary and Shadow Event Processors up; Event Server up.
32	Primary and Shadow Event Processors up; Primary and Dual-Event Servers.
50	Event Processor is "not running" because it could not connect to the machine in the EDMachines list in the AutoSys configuration file; No Event.
51	Event Processor "not running" (see return code 50); Event Server up.
52	Event Processor "not running" (see return code 50); Primary and Dual-Event Servers up.



Instructor
Notes

Return Code	Optional Message
60	Event Processor “not running” because no machines listed in the EDMachines list in the AutoSys configuration file; No Event Server.
61	Event Processor “not running” (see return code 60); Event Server up.
62	Event Processor “not running” (see return code 60); Primary and Dual-Event Servers up.
99	One of the following can cause this message to appear: <ul style="list-style-type: none">■ One or more of the following environment variables is not set or is set incorrectly: AUTOSYS, AUTOSERV, AUTOUSER.■ You issued the <code>chk_auto_up</code> command with invalid arguments.



Skill Practice

Task Purpose: Determine if the Event Server and the Event Processor are running.

- 1 Run `chk_auto_up`.
- 2 Discuss the output.

Task Summary

You now know how to confirm that the Event Processor and Event Server are actually running by using the `chk_auto_up` command.



Skill Practice

When running `chk_auto_up`:

- `ECHOS$? = UNIX`
- `ECHO%ERRORLEVEL% = NT`

Close

the task by reviewing concepts and checking understanding.

Instructor
Notes

Slide 4-8



Task 4: View Current Version of Unicenter AutoSys JM

To print information about Unicenter AutoSys JM and the system configuration, use the `autoflags` command.

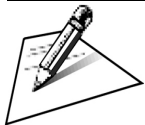
Slide 4-9



Syntax

`autoflags [-a | -i | -o | -d | -v | -r | -h | -n]`

Option	Displays
<code>-a</code>	All autoflags information to standard output.
<code>-i</code>	The tape ID number to standard output.
<code>-o</code>	The operating system to standard output.
<code>-d</code>	The database type to standard output, either SYB for Sybase or ORA for Oracle.
<code>-v</code>	The version number to standard output.
<code>-r</code>	The release number to standard output.
<code>-h</code>	The host ID to standard output.
<code>-n</code>	The host name to standard output.



Instructor
Notes

Slide Show slide 8 to introduce the Task.

Slide Show slide 9 to discuss command syntax.



Skill Practice

Task Purpose: View the current version of Unicenter AutoSys JM.

Type the following at the prompt:

```
autoflags -a
```

Task Summary

You now know how to use the `autoflags` command to view current system configuration.



**Review and
close**

the task. Discuss the Skill Practice and check students' understanding.

Instructor
Notes

Slide 4-10



Task 5: Report Job Dependencies

To report information about the dependencies and conditions of a specific job, use the `job_depends` command.

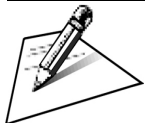
Slide 4-11



Syntax

```
job_depends [-c | -d | -t] [-J job_name] [-F from_date/time]
            [-T to_date/time] [-L print_level] [-D data_server:database |
            -D TNSname]
```

Parameter	Description
-c	Prints out the current state of a job and the names of any jobs that are dependent on this job.
-d	Prints out the starting conditions for a job; no indication of the current status of the job is provided. For box jobs, jobs inside the box are shown hierarchically.
-t	Prints out the starting conditions for a job; however, the top level of jobs (or boxes) that are reported are limited to those that will start within the time period specified by the job or box's date conditions.
-J job_name	Indicates the job on which to report, where <code>job_name</code> is the name of the target job. To report on all jobs, enter the word ALL for the <code>job_name</code> . You can also use the % wildcard.



Slide Show slide 10 to introduce the Task.

Slide Show slide 11 to discuss command syntax.

Instructor
Notes

Parameter	Description
-F from date/time	Indicates the report start date and time, where <code>from_date/time</code> is the date and time of the first job in the report. This option is used with the -T option only. The format is MM/DD/[YY]YY HH:MM.
-T to date/time	Indicates the report end date and time, where <code>to_date/time</code> is the date and time of the last job in the report. This option is used with the -F option only. The format is MM/DD/[YY]YY HH:MM. If this option is not specified, <code>job_depends</code> will search without limitation into the future.
-L print_level	Indicates the print level for the report, where <code>print_level</code> is any valid numeric value specifying the number of levels of nesting to display for a box job. This option is used with the -d and -t options only.
-D data_server:database	Indicates the name of the Sybase or Microsoft® SQL Server™ data server, and the specific database within it, to be searched for the specified information.
-D TNSname	Indicates the TNS alias name of the Oracle data server to be searched for the specified information. Normally, <code>job_depends</code> consults the environment variables and the Administrator configuration settings to determine to which database to connect. This option enables <code>job_depends</code> to report on any Unicenter AutoSys JM Event Server on the network.



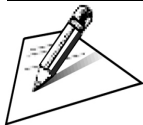
Instructor
Notes

Parameter	Description
-D data_server: database	Indicates the name of the Sybase data server, and the specific database within it, to be searched for the specified information. Normally, <code>autorep</code> consults the configuration file (<code>\$AUTOUSER/config.\$AUTOSERV</code>) to determine to which database to connect. This option enables <code>autorep</code> to report on any Event Server on the network.
-D TNSname	Indicates the TNS alias name of the Oracle data server to be searched for the specified information. Normally, <code>autorep</code> consults the configuration file (<code>\$AUTOUSER/config.\$AUTOSERV</code>) on UNIX, or the Windows NT Registry, to determine to which database to connect. This option enables <code>autorep</code> to report on any Event Server on the network.

Example

Consider the paycheck job example used earlier; you wish to display a report on its current condition. The job is called `job_pay`. You issue the following command:

```
job_depends -c -J job_pay
```



Instructor
Notes

■ Use Basic Commands

Example

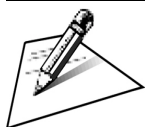
You would see a report similar to the following, displayed to standard output:

```
Start Dependent
Job Name Status Date Cond? Cond? Jobs?
-----
job_pay INACTIVE No      No      Yes
Dependent Job Name
-----
jobY
```

This report shows that the paycheck job has no date or starting conditions, but another job, j obY is dependent on it.

Task Summary

You now know how to use the j ob_depends command to list the dependencies of specific jobs. The next Task will cover the command used to view the Event Processor and Remote Agent logs.



Instructor
Notes

Review	the example provided with additional options; discuss output.
Close	the task by reviewing concepts and checking for understanding.

Slide 4-12



Task 6: View the Event Processor Log or Remote Agent Log

To display the Event Processor and Remote Agent log files, use the `autosyslog -e` command.

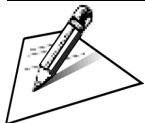
Slide 4-13



Syntax

```
autosyslog [-e | -J job_name] [-p]
```

Option	Description
-e	Indicates that the Event Processor log is to be monitored. <i>When in this mode, in order to terminate the command, you must press Ctrl+C.</i>
-J job_name	Indicates that the Remote Agent log for the specified <code>job_name</code> is to be viewed. Note • You must issue this command on the machine where <code>job_name</code> ran.
-p	The -p option must be used with the -J <code>job_name</code> option. Use this to specify profile information.



- Slide** Show slide 12 to introduce the Task.
- Slide** Show slide 13 to discuss command syntax.

Instructor
Notes

Example

Example

To view the Remote Agent log of the last run of the paycheck job, you would issue the following command on the command line of the machine where the paycheck job ran:

```
autosyslog -J job_pay
```

Task Summary

You now know how to view the logs of specific jobs using the `autosyslog` command. In the next Task, you will learn how to send events from the command line.



Instructor
Notes

Close the task by reviewing key concepts within the context of the example provided. Do “what if” scenarios, if time permits, to illustrate the various possibilities. Check for understanding.

Slide 4-14



Task 7: Send Events

To send events for a variety of purposes, including starting or stopping jobs, stopping the Event Processor, and putting a job on hold, use the `sendevent` command. This command is also used to set global variables or cancel a scheduled event.

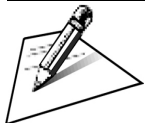
Slide 4-15



Syntax

```
sendevent -E event [-S autoserv_instance] [-A alarm] [-J job_name]
[-s status] [-C comment] [-P priority] [-M max_send_trys]
[-q job_queue_priority] [-T "time_of_event"]
[-G "global_name=value"] [-k signal_numbers] [-u]
```

Option	Description
-E event	Specifies the event to be sent.
-S autoserv_instance	Specifies the three-character instance, for example, U45, to which the event should be sent. If not specified, <code>sendevent</code> will use the value of the environment variable named <code>%AUTOSERV%</code> .
-A alarm	Specifies the name of the alarm to be sent.
-J job_name	Specifies the name of the job to which the specified event should be sent.
-s status	Specifies the status to which the job specified in <code>-J job_name</code> should be changed.



Slide Show slide 14 to introduce the Task.

Slide Show slide 15 to discuss command syntax.

Instructor
Notes

■ Use Basic Commands

Syntax

Option	Description
-C <i>comment</i>	Specifies a textual comment that is to be attached to this event for documentation purposes only.
-P <i>priority</i>	Specifies the priority to be assigned to the event being sent.
-M <i>max_send_trys</i>	Specifies the maximum number of times <code>sendevent</code> will attempt to send the event to the database.
-q <i>job_queue_priority</i>	Specifies the new queue priority to be assigned to the job.
-T " <i>time_of_event</i> "	Specifies the date and time when the event should be processed. The format is MM/DD/[YY]YY hh:mm, where hh denotes hours and must be from 00 to 23. Double quotes are required as part of the specification. This is used to schedule an event in the future. The default is to process the event immediately.
-G " <i>global_name=value</i> "	Specifies the name and value of a global variable when a SET_GLOBAL event is sent. The <i>global_name</i> and the value can each be a maximum of 30 characters (leading and trailing spaces in the value are ignored).
-k <i>signal numbers</i>	For processes running on UNIX, this argument specifies the signal number.
-u	Cancels the event specified in the -E event option. This option can be used only for unprocessed events that are scheduled to be processed in the future.



Instructor
Notes

Examples

To start a job named `job_sales` that has no starting conditions (and therefore must be started manually), type:

```
sendevent -J job_sales -E STARTJOB
```

To force start a job named `job_daily`, which is waiting on the completion of another job, and explain the reasons for your action, type:

```
sendevent -J job_daily -E FORCE_STARTJOB -C "wanted to leave  
early, forced it"
```

To set a global variable named `approval`, to a value of "Manager_J_Doe_OK", type:

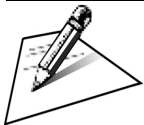
```
sendevent -E SET_GLOBAL -G "approval=Manager_J_Doe_OK"
```

To cancel all unprocessed `JOB_OFF_HOLD` events for a job named `rundata`, type:

```
sendevent -E JOB_OFF_HOLD -J rundata -u
```

Task Summary

You now know how to use the command line to send events.



Instructor
Notes

Close

the task by reviewing concepts within the context of the examples provided. Check for students' understanding.

Skill Builder: Discuss Command Usage



Business Problems

For each of the following bullet points, discuss what command and options you should use to solve the stated problem.

- RBC's production web server periodically receives alerts because thresholds are running too high. You need to know how many jobs are running at any given time.
- If the production web server should become unreachable while it is running a job, you want to detect that the machine is down, receive an alarm, and notify the job owner.
- RBC's week-ending job failed to run as scheduled. You must determine why.
- RBC needs to determine which version of Unicenter AutoSys JM is currently running.
- One of the jobs affected by the production web server issue was a box job that was just terminated. Before restarting it, you want to examine the logic of all jobs contained in the box, and verify their current statuses.
- You have examined the jobs contained in the box job and verified their statuses. You just restarted the job, but now want to monitor the statuses as the box runs.
- You are short-handed today and another operator just went home sick. To handle the workload, you need to start a job sooner than scheduled and explain why.



Instructor
Notes

Solution to Skill Builder Discussion

autorep
chase
chk_auto_up
autoflags
job_depends
autosyslog -e
sendevent Force Start Job with comment - followed by Job on Hold

Slide 4-16

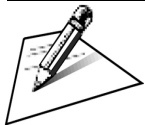


Module Summary

You should now be able to:

- View a Job Detail Report for Multiple Jobs
- View Chase Output for All Running Jobs
- Verify that the Event Processor and Event Server are Running
- View Current Version of Unicenter AutoSys JM
- Report Job Dependencies
- View the Event Processor Log or Remote Agent Log
- Send Events

This module presented some basic commands for managing jobs. You can check the *Unicenter AutoSys JM Reference Guide for UNIX and Windows* for a complete command description. In the following Modules, you will learn to schedule jobs that depend on another job or on a specific date or time.



Instructor
Notes

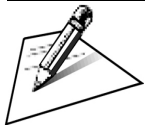
Slide Show slide 16 and review Module Summary.
Check students' understanding.

■ Use Basic Commands

Module Summary



Instructor
Notes



Instructor
Notes

■ Use Basic Commands

Module Summary



Instructor
Notes

Schedule Jobs

Slide 5-1



Module Objectives

After this module, you will be able to:

- Create and Test Jobs Dependent on Job Status
- Submit a Date/Time Schedule
- Submit Alarms/Terminators
- Specify Additional Job Attributes

Module Overview

RBC's internal training team wishes to push courseware to students registered for a specific training course. The contact information for each registered student is stored in a file that must be merged with the training materials. If the job should fail, you would like to have it automatically restart and notify an operator. When it succeeds, it triggers another job to print out completion certificates for the instructor to distribute in class.

In this Module, you will learn how create jobs that depend on other jobs to start, or on specific dates and times. You will also learn how to set jobs to notify you when a specific status is obtained. In the following Module, you will extend this foundation to creating custom calendars.



Instructor
Notes

Slide Show slide 1 to introduce Module Objectives.
Review the Problem described in the Overview to tell students "what's in it for me?"

Slide 5-2



Task 1: Create and Test Jobs Dependent on Job Status

Term	Definition
<i>dependent</i>	JobB is <i>dependent</i> on JobA; how it is dependent varies, but regardless of the type of dependency, JobB cannot run until JobA fulfills the condition.

Syntax

To express a job dependency, use the following format in the dependency field of the Job Editor or Job Definition tool:

status(job_name)

where *status* represents any status listed in left column of the following table. You may spell out the status or abbreviate it:

Dependency Type	Syntax and Description
SUCCESS	JobB s (JobA) or JobB success (JobA) JobB depends on the SUCCESS of JobA to execute.
FAILURE	JobB f (JobA) JobB depends on the failure of JobA to execute.



Instructor
Notes

Slide Show slide 2 to introduce the Task.
Define terms.
Discuss syntax.

Dependency Type	Syntax and Description
TERMINATED	<code>JobB t (JobA)</code> The status of JobA must show TERMINATED for JobB to execute.
NOT RUNNING	<code>JobB n (JobA)</code> The status of JobA must show NOT RUNNING for JobB to execute.
DONE	<code>JobB d (JobA)</code> The status of JobA must show SUCCESS, FAILURE, or TERMINATED for JobB to execute.
EXIT CODE	<code>JobB e (JobA)=0</code> JobB executes after JobA returns an EXIT CODE of 0.

Success Dependencies

A SUCCESS dependency means JobB depends on the success of JobA to run.

Note • Job names are case-sensitive. "JobA" is not identical to "Joba."

Syntax

`s (job_name)`

where s represents a status of SUCCESS for the specified job.



Instructor
Notes

Examples of a SUCCESS Dependency

Consider a business situation in which sales bonuses are based on the combined sales posted for all members of a given business unit.

Let JobA represent the job that posts all sales data.

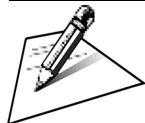
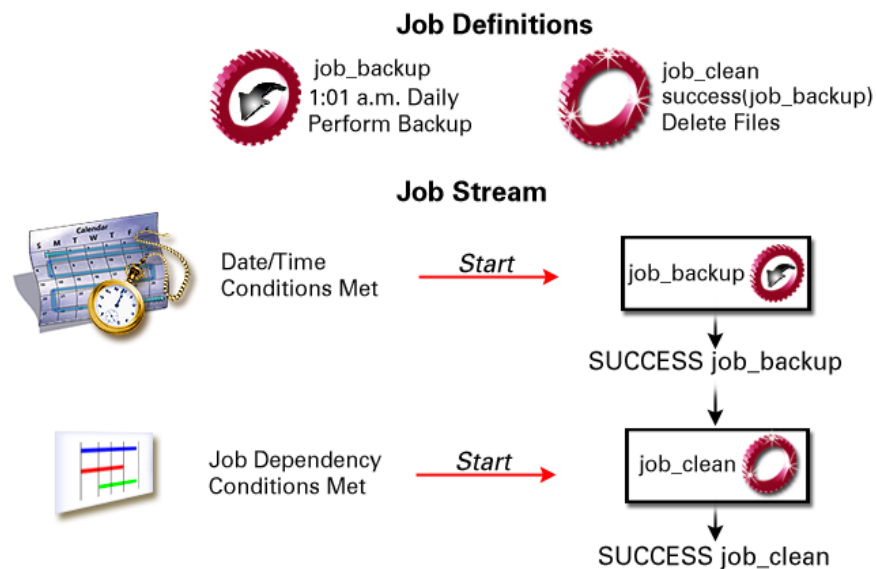
Let JobB represent the job that calculates the bonus due on the combined total.

We cannot calculate the bonus due until we first total all of the sales. Therefore, JobB depends on the successful completion of JobA, expressed in the syntax `jobB s(jobA)`.

Slide 5-3



The following graphic illustrates another SUCCESS dependency:



Instructor
Notes

Slide Show slide 3 to illustrate dependencies in a jobstream.
Discuss what is happening in the graphic, emphasizing that dependencies give Unicenter AutoSys JM its power to automate complex operations.

The job called `job_backup` is scheduled to run every day at 1:01 AM. It is a command job that executes a batch program, which physically backs up a specific drive, on a client machine.

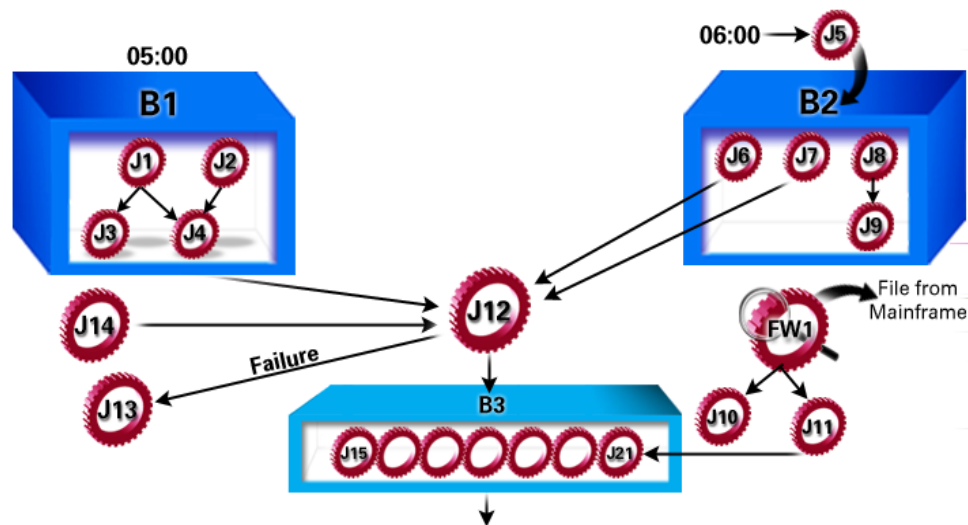
The job called `job_clean` deletes all of the files off the drive and must only be started when the backup job is successful, to prevent files from being deleted before being backed up.

Slide 5-4



Discussion

You can continue to define jobs dependent on other jobs to form complex job streams, like this:



The arrows all represent SUCCESS, unless otherwise stated.

At 5 AM, box job B1 is started. Within it, job J1's successful completion will start J3, but only when jobs J1 and J2 both successfully complete, will J4 start. Meanwhile, at 6 AM, job J5 runs. When it completes, box B2 is started. Within box B2, jobs J6, J7, and J8 are started. Together with box B1, jobs J6,



Instructor
Notes

Slide Show slide 4 to discuss a complicated jobstream. Explain what is happening in the graphic.

J 7, and J 14 prompt the start of job J 12. If any of the specified jobs should not run, job J 12 will not start. If Job J 12 should fail, only then is job J 13 started. Finally, when Job J 12 completes successfully, box B 3 starts. But box B 3 cannot complete until something else happens. What must happen for box B 3 to complete?

Note • Continue the following job naming convention as you perform the Interactive Demonstration:

name_[u|w]_commandname



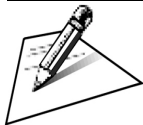
Interactive Demonstration

Task Purpose: Create a job in UNIX that is dependent upon the success of another job to start. You must create two jobs to set up such a dependency according to the syntax presented: `dep job s (name_u_commandsuccess)`

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type a job name:

name_u_commandsuccess
- 3 In the **Job Type** field, click **Command**.
- 4 In the **Execute on Machine** field, type the name of the machine on which the command job will run. You should enter your own valid, licensed client machine.

Note • Your instructor will supply the appropriate machine name.



Instructor
Notes

Discussion

To answer the question at the end of the Discussion example, direct your students' attention to the file watcher job FW1. Job J11 depends on the successful completion of the file watcher job. In turn, Job J21 depends on the successful completion of Job J11. So, the box B3 cannot complete until J11 is successful.

- 5 In the **Command to Execute** field, type the command that is to run when the job starts, such as:

```
echo hello
```

Note • Although you could redirect the output of this command to a file directly in this field, we suggest you click **Adv. Features** and complete the attribute called **File to Redirect Standard Output**, as follows:

```
/tmp/name_u_commandsuccess.out
```

- 6 Click **Save** to save the job definition for `name_u_commandsuccess`.
- 7 Create a new job definition called:

```
name_u_depjob1
```

- 8 In the **Job Type** field, click **Command**.

- 9 In the **Command** field, type:

```
sleep 60
```

- 10 In the **Execute on Machine** field, type the name of the machine on which the command job will run. You should enter your own valid, licensed client machine.

Note • Your instructor will supply the appropriate machine name.

- 11 Under the **Starting Parameters** section, in the **Starting Condition** field, type:

```
s(name_u_commandsuccess)
```

- 12 Save `name_u_depjob1` and run `name_u_commandsuccess`.



Instructor
Notes

- 13 Click **OK** in response to the message box. The job is successfully created. To test the dependency you just created:
- From the Unicenter AutoSys JM toolbar, click **Job Activity Console**.
 - Right-click *name_u_commandsuccess*.
 - From the shortcut menu, choose **Start Job** and click **Yes** in response to the message box.
 - From the **Job Activity Console**, check the Status field. Both jobs should be marked **SUCCESS**. Even though you did not start the depjob job manually, it still ran because it was set to run upon the successful completion of *name_u_commandsuccess*.

Discussion

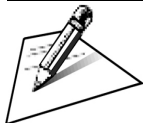
Suppose you wish for something else to happen — such as, print the file — after your depjob file watcher finds the *name_u_commandsuccess.out* file. What is your next step?

Interactive Demonstration



Task Purpose: Create a job called depjob1 that is dependent upon the success of another job to start. You must create two jobs (a JobA and a JobB) to show the dependency.

- From the Unicenter AutoSys JM toolbar, click **Job Editor**.
- From the **Autosys Job Editor** dialog box, choose **File ▶ New**.
- Type an appropriate name. For this example, type the following and click **OK**:
name_w_commandsuccess



Solution for Discussion

Have students discuss possible solutions.

Solution: create a job that depends on the success of depjob.

Instructor
Notes

■ Schedule Jobs

Interactive Demonstration

- 4 In the **Command** field, type an appropriate command for the job to execute. For this example, use:

```
echo hello
```

Note • You should not redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Command Info, as follows:

```
c:\temp\name_w_commandsuccess.out
```

- 5 In the **Send to Machine** field, select your *machinename*. In practice, you will use the name of your Event Processor machine.
- 6 Choose **File ▶ Save**, click **OK**, and click **OK** again to clear the message boxes.
- 7 Choose **File ▶ New** to create a second job that will depend on *name_w_commandsuccess*.
- 8 In the **New Job** dialog box, type the following and then click **OK**:

```
name_w_depjob1
```

- 9 In the **Job Type** field, select **Command**.
- 10 In the **Command** field, type: echo hello

Note • You should not redirect the output of this command directly in the command field, instead, you must use the attribute called File to Redirect Standard Output in Command Info, as follows:

```
c:\temp\name_w_depjob1.out
```



Instructor
Notes

11 In the **Dependencies** field, type:

`s (name_w_commandsuccess)`

This syntax is interpreted as "Success of *name_w_commandsuccess*."

12 In the **Send to Machine** field, select your *machinename*.

13 Choose **File ► Save**, and click **OK**.

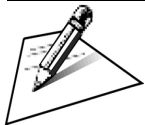
14 Click **OK** in response to the message box. The job is successfully created.

15 To test the dependency you just created:

- a From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.
- b Right-click *name_w_commandsuccess* and choose **Start Job**.
- c Click **Yes** in response to the message box.
- d From the **Scheduler Console**, view the Status field. Both jobs should be marked **SUCCESS**. Even though you did not start the *dep job* manually, it still ran because it was set to run upon the successful completion of *name_w_commandsuccess*.

Failure Dependencies

A FAILURE dependency works much like a SUCCESS dependency, except you schedule one job to run only when another job fails. For example, you may need to run a recovery job after a processing job fails.



Instructor
Notes

Syntax

`f (job_name)`

where `f` represents FAILURE, and is read as "... the failure of the specified job." (Alternatively, you may spell out the whole word.)

Example

If you run a job and it fails, you may want to generate a report on the failure.

Returning again to the previous example in which bonuses are calculated based on the combined sales totals of the entire business unit, assume the following:

JobA runs the batch job that prints the sales commissions paid to each member of the unit. The job fails if the printer is not loaded with blank check stock.

JobB runs the job that notifies an operator when the printer requires restocking.

The syntax is therefore:

`f (JobA)`

Interactive Demonstration



Task Purpose: Modify an existing job in UNIX so that it is dependent on another job's failure to execute. In this exercise, you will set `depjob` to start only if `name_u_commandfail` returns a FAILURE status.

- 1 In the **Job Name** field, type the following to list all stored jobs and then click **Search: %**
- 2 Double-click `name_u_depjob1` to select it.
- 3 Under the Starting Parameters section, in the **Starting Condition** field, change the dependency string to: `f (name_u_commandfail)`



Instructor
Notes

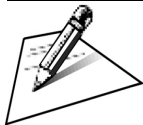
- 4 Save the job.
- 5 Close the **Job Definition** tool.
- 6 Start *name_u_commandfail*.
- 7 Click **Yes** to clear the message box.
- 8 Watch the Event Report section of the **Job Activity Console** for status indicators. The job called *name_u_commandfail* reads FAILURE because it cannot process the syntax error you typed in Module 2, which then triggers *name_u_depjob1*.



Interactive Demonstration

Task Purpose: Modify an existing job in Windows so that it is dependent on another job's failure to execute. In this exercise, you will set *depjob* to start only if *name_w_commandfail* returns a FAILURE status.

- 1 To modify *name_w_depjob1* to be dependent on the failure of *name_w_commandfail*, click **Scheduler Console** from the Unicenter AutoSys JM toolbar.
- 2 Right-click *name_w_depjob1* and choose **Job Editor** from the shortcut menu.
- 3 In the **Dependencies** field, change the dependency string to:
`f (name_w_commandfail)`
- 4 Choose **File ► Save** and click **OK**.
- 5 Choose **File ► Exit** to close the **Job Editor** dialog box.
- 6 Right-click *name_w_commandfail* and choose **Start Job** from the shortcut menu.
- 7 Click **Yes** to clear the message box.



Instructor
Notes

- 8 Watch the **Scheduler Console** for status indicators. The job called *name_w_commandfail* reads FAILURE because it cannot process the syntax error we typed in Module 2, which then triggers *name_w_depjob1*.

Terminated Dependencies

Jobs may terminate for various reasons: they may have passed the maximum permitted run time, a user may have sent a KILLJOB event, or they may have been set to terminate if the box containing it failed. However, if a job failed, it has a FAILURE status, not TERMINATE.

You may wish to do something, print a report, back up a directory, and so on, when a job is terminated. For example, even though Unicenter AutoSys JM sends an alarm, you may need to contact your boss whenever the commission payment job is terminated. You would create a job that pages your boss when the status of the commission payment job is TERMINATED.

Syntax

`t(job_name)`

where `t` represents a status of TERMINATED for the specified job.

Interactive Demonstration



Task Purpose: Modify an existing job in UNIX to execute after another job is terminated. In this example, make *depjob* dependent on the TERMINATE status of *name_commandterminate*.

- 1 Create a new command job and name it:

name_u_commandterminate



Instructor
Notes

- 2 In the **Command** field, type a command you can use to force the TERMINATE status, such as:

sleep 90
- 3 Save the job.
- 4 Modify *name_u_depjob1* so that it is dependent on the termination of *name_u_commandterminate*:

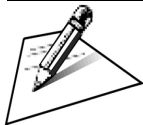
t(*name_u_commandterminate*)
- 5 Save *name_u_depjob1*.
- 6 Start *name_u_commandterminate*.
- 7 From the **Job Activity Console**, select the *name_u_commandterminate* job and click **Kill Job**. A confirmation message is displayed; click **Yes** to continue.
- 8 Watch the Event Report section of the Job Activity console for status updates. Your *name_u_depjob1* job should start when *name_u_commandterminate* terminates.

Interactive Demonstration



Task Purpose: Modify an existing job in Windows to execute after another job is terminated. In this example, make depjob dependent on the TERMINATE status of *name_u_commandterminate*.

- 1 Open **Job Editor**.
- 2 In the **Job Name** field, type: *name_w_commandterminate*



Instructor
Notes

- 3 In the **Command** field, type a command we can use to force the TERMINATE status, such as:

notepad.exe

When this job executes, Notepad will open and you will terminate the program using Send Event. That will force the dependent job to execute.

- 4 Choose **File ► Save** and close **Job Editor**.
- 5 From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.
- 6 Right-click *name_w_depjob1* and choose **Job Editor** from the shortcut menu.
- 7 In the **Dependencies** field, change the dependency string to:
`t(name_w_commandterminate)`

Note • Watch the bottom of the dialog box for syntax error help.

- 8 Choose **File ► Save** and click **OK**.
- 9 Choose **File ► Exit** to close the **Job Editor** dialog box.
- 10 Right-click *name_w_commandterminate* and choose **Start Job** from the shortcut menu.
- 11 Click **Yes**.
- 12 Watch the status column, which should show the status of *name_w_commandterminate* as **RUNNING**.
- 13 From the **Scheduler Console**, click **Send Event** to open the Send Event tool.
- 14 Under the **Event Type** group, select the **Change Status** option.
- 15 In the **Status** field at the bottom of the tool, select **Terminated** from the list.



Instructor
Notes

This demonstration illustrated the earlier discussion showing that even though the job was given a TERMINATED status; however, the Notepad is still running. Because the job was not really killed, using the Send Job Event.

16 Click **Send** and click **OK** to clear the message box.

You see a status of **TERMINATED** on the console. The **Scheduler Console** should now show *name_w_depjob1* starting and then running (**SUCCESS**).

Note • Check the Last Change Time in the Scheduler Console as another way to verify a job did start.

Not Running Dependencies

A status of **NOTRUNNING** means the status of a job is *anything* except **RUNNING**. You might want to create such a dependency whenever running two specific jobs simultaneously would render your system unavailable for too many hours — for example, running a back-up job at the same time a large quarterly report job is scheduled.

Syntax

`n(job_name)`

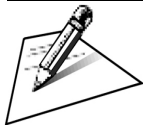
where *n* represents a status of **NOTRUNNING** for the specified job.

Interactive Demonstration



Task Purpose: Modify an existing job in UNIX to run only when another job is not running — *depjob* and *name_commandsuccess* cannot run simultaneously or we risk tying up the system for hours.

- 1 Modify *name_u_depjob1* so that it is set to the following dependency:
`n(name_u_commandsuccess)`
- 2 Save *name_u_depjob1*.
- 3 Run *name_u_commandsuccess* and watch the log for progress updates.



Instructor
Notes



Interactive Demonstration

Task Purpose: Modify an existing job in Windows to run only when another job is not running — *name_u_depjob1* and *name_u_commandsuccess* cannot run simultaneously or we risk tying up the system for hours.

- 1 To modify the *depjob1* job for this example, click **Scheduler Console** from the Unicenter AutoSys JM toolbar.
- 2 Right-click the *depjob1* job and choose **Job Editor**.
- 3 Modify the **Dependencies** field to read: *n(name_w_commandsuccess)*, where *n* is NOTRUNNING.
- 4 Choose **File ▶ Save** and click **OK**.
- 5 Choose **File ▶ Exit** to close the Job Editor.
- 6 Right-click *name_w_commandsuccess* and choose **Start Job**.
- 7 Click **Yes**.
- 8 Watch the status column, which should list a status of RUNNING. Notepad should open. Note the last change time for the *depjob1* job. When you close Notepad, the *name_w_commandsuccess* status should update to SUCCESS.
- 9 Click **Start ▶ Programs ▶ Computer Associates ▶ Unicenter ▶ AutoSys Job Management (W45) ▶ Command Prompt**, to open the AutoSys Log.
- 10 Watch the AutoSys Log messages for the *depjob1* job. *depjob1* will not execute because the status of *name_w_commandsuccess* is RUNNING, satisfying the dependent condition. As soon as you end the *name_w_commandsuccess* job, the *depjob1* job will be executed, because the condition has been satisfied.



Instructor
Notes

Done Dependencies

A condition status of DONE means the job status is SUCCESS, FAILURE, or TERMINATED, allowing a single status to represent three results, simplifying your work.

Syntax

`d(job_name)`

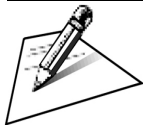
where `d` represents a status of DONE for the specified job.



Interactive Demonstration

Task Purpose: Modify an existing job to run when some other job results in SUCCESS, FAILURE, or TERMINATED.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type `name_u_depjob1`.
- 3 Click **Search**. The Job Definition parameters for the specified job appear.
- 4 Change the **Starting Condition** to:
`d(name_u_commandsuccess)`
- 5 Save `name_u_depjob1`.
- 6 Run `name_u_commandsuccess`.



Instructor
Notes



Interactive Demonstration

Task Purpose: Modify an existing job to run when some other job results in SUCCESS, FAILURE, or TERMINATED.

- 1 From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.
- 2 Right-click your depjob1 job and choose **Job Editor**.
- 3 Modify the **Dependencies** field to read: `d(name_w_commandsuccess)`, where d is DONE.
- 4 Choose **File ▶ Save** and click **OK**.
- 5 Choose **File ▶ Exit**.
- 6 Right-click *name_w_commandsuccess* and choose **Start Job** from the shortcut menu.
- 7 Click **Yes**.
- 8 When Notepad opens, close it. That will start depjob. Note the **last change time**. It should be later than the *name_w_commandsuccess* last change time, which proves the job ran after *name_w_commandsuccess* is DONE. This does *not* mean it was successful, or that it failed; it means only that the depjob job ran at least once and stopped.

Note • *DONE* = SUCCESS, FAILURE, or TERMINATED.
NOTRUNNING = Any status except RUNNING.



Instructor
Notes

Exit Code Dependencies

Programs return an *exit value* that is programmed within the executable code. This exit value is the last thing returned when the program terminates. Generally, a zero exit code indicates *success*, while a non-zero exit code indicates an error. The expected error values should be documented with each individual program, but some programs can return unexpected exit codes. You should modify these programs so that they return expected values. Use these values when specifying exit code dependencies.

For example, if a broken communication line results in JobA failing with an exit code of 4, when this code is encountered, you want the system to execute a shell script called JobB to redial the line.

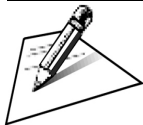
Syntax

e(job_name) operator value

where *e* represents a status of EXITCODE for the specified job,
where *operator* is one of the following,

- = equal to *value*
- != not equal to *value*
- < less than *value*
- <= less than or equal to *value*
- > greater than *value*
- >= greater than or equal to *value*

and *value* is any integer. A *value* of 0 is the default for a SUCCESS status.



Instructor
Notes



Interactive Demonstration

Task Purpose: Modify an existing job to start when another job returns the specified exit code.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type the following and then click **Search**:
`name_u_depjob1`
- 3 Modify the **Starting Condition** field to read:
`e(name_u_commandsuccess)=0`
where e is EXITCODE and 0 = success.
- 4 Save the `name_u_depjob1` job.
- 5 Start `name_u_commandsuccess`. Watch the Event Report section for status updates.



Interactive Demonstration

Task Purpose: Modify an existing job to start when another job returns the specified exit code.

- 1 From the Unicenter AutoSys JM toolbar, click **Scheduler Console**.
- 2 Right-click the `name_w_depjob1` job and choose **Job Editor**.
- 3 Modify the **Dependencies** field to read:
`e(name_w_commandsuccess)=0`
where e is EXITCODE and 0 = success.



Instructor
Notes

- 4 Choose **File ▶ Save** and click **OK**.
- 5 Choose **File ▶ Exit** to close the Job Editor.
- 6 Right-click `name_w_commandsuccess` and choose **Start Job** from the shortcut menu.
- 7 Click **Yes**.
- 8 When Notepad opens, close it. That will start the `depjob` job. Note the **last change time**, which should be after the `name_w_commandsuccess` last change time, proving `name_w_commandsuccess exit code=0`.

Use the *And, Or, and ()* Operators

You can configure complex conditions by combining a series of conditions with the AND and the OR logical operators. You can enter these operators in upper- or lowercase, but not in mixed case. In addition, you can use the pipe symbol (|) instead of the word OR, and the ampersand symbol (&) instead of the word AND. Spaces between conditions and delimiters are optional.

You can specify more complex conditions by grouping the expressions in parentheses. The parentheses do not imply any sort of precedence; they are simply used for grouping.

Syntax

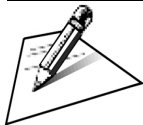
```
(s(JobA) AND s(JobB)) OR (d(JobD) AND d(JobE))
```

which means:

a job depends on the success of *both* JobA and JobB to run, *or* on the status of DONE for *both* JobD and JobE.

The same example could also be written as:

```
(s(JobA)&s(JobB)) |(d(JobD)&d(JobE))
```



Instructor
Notes



Interactive Demonstration

Task Purpose: Modify an existing job that is dependent on *both* of the following conditions to start: the success of `commandsuccess` and the failure of `commandfail` or the success of `commandfail` and done of `commandsuccess`.

- 1 Define the dependent job:
 - a From the Unicenter AutoSys JM toolbar, click **Job Definition**.
 - b In the **Job Name** field, type: `name_u_depjob1`
The job definition for the specified job appears.
 - c Modify the **Dependencies** field to:
`s(name_u_commandsuccess) AND f(name_u_commandfail)`
 - d Save `name_u_depjob1`.
- 2 Start your `commandsuccess` job and then watch the Event Report section of the Job Activity Console for status updates.
Did your `depjob1` job start?
- 3 Start your `commandfail` job and then watch the Event Report section of the Job Activity Console for status updates.
Did your `depjob1` job start now?



Interactive Demonstration

Task Purpose: Modify one existing job to force a SUCCESS status, a second existing job to force a FAILURE status, and to create a third job that is dependent on *both* of the previous jobs to start.

- 1 Right-click the `depjob1` job and choose **Job Editor**.



Instructor
Notes

- 2 Modify the **Dependencies** field to reflect:
`s(name_w_commandsuccess) AND f(name_w_commandfail)`
- 3 Choose **File ▶ Save** and click **OK**.
- 4 Choose **File ▶ Exit**.
- 5 Right-click *name_w_commandsuccess* and choose **Start Job**.
- 6 Click **Yes**.
- 7 Check the **last change time** for the depjob1 job. It has not yet run.
- 8 Right click *name_w_commandfail* and choose **Start Job**.
- 9 Click **Yes**.
- 10 Check the **last change time** for the depjob1 job; this time, it did run.

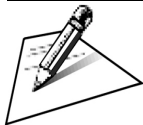


Skill Practice

Task Purpose: Modify one existing job to force a SUCCESS status, a second existing job to force a FAILURE status, and to create a third job that is dependent on one of the previous jobs to execute. You will use the OR operator in this exercise.

- 1 Modify your depjob1 dependency to be dependent on the success of your `commandsuccess` job *or* the failure of your `commandfail` job.
- 2 Save your depjob1 job.
- 3 Start your `commandsuccess` job.
- 4 Watch the Event Report section for status updates.

Did your depjob1 job run?



Instructor
Notes

Discuss potential scenarios for using complex dependent conditions. List ideas on the white board.

Task Summary

You now know what job dependencies are, and how to use them to manage your jobs and create complex jobstreams. In the next Task, you will learn how to define jobs that start on specific dates or times.



Instructor
Notes

Close the Task by reviewing concepts and checking understanding.

Slide 5-5



Task 2: Submit a Date/Time Schedule

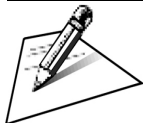
You can create time-based start conditions for Unicenter AutoSys JM jobs. Time-based job starts can be set for any day of the week or for a specific date. To specify exact dates, you will use the Calendar feature. Within each day or date, you can specify a time, but the time range cannot span more than 24 hours.

Time-based job starts are set using the **Date/Time** tab on the Job Editor in Windows. In UNIX, you use the **Date/Time Options** button on the Job Definition tool to access a similar set of parameters.

Note • The UNIX Date/Time Options button is not available until you click Yes next to Is the Start Date/Time Dependent?

The following table describes the various options and fields available:

GUI Element	Description
Date/Time Conditions check box	<p>This box enables all of the date/time settings on the tab.</p> <p>Note • In UNIX, the Yes option next to the Is the Start Date/Time Dependent question functions identically.</p>
Run Calendar	<p>This option permits you to select a previously-created Calendar from the list. You use the Calendar Editor to create custom calendars, such as the due dates for certain bills.</p> <p>Note • In UNIX, clicking the Calendars button located at the bottom left corner of the Date/Time Options dialog accomplishes the same objective.</p>



Instructor
Notes

Slide Show slide 5 to introduce the Task.
Define Terms

■ Schedule Jobs

Task 2: Submit a Date/Time Schedule

GUI Element	Description
Run Days	<p>This option permits you to schedule any day of the week. The All and None buttons within the Run Days group select or clear all days, respectively.</p> <p>Note • In UNIX, you directly select days by clicking Every Day or a specific day.</p>
Exclude Calendar check box	<p>This box is similar to Run Calendar; it permits you to select a previously-created calendar of dates to exclude, such as the corporate holidays for each of your international offices.</p> <p>Note • In UNIX, the Do NOT Run on Days in Calendar (Exclude) parameter offers a text field in which you can type or search for an existing calendar name.</p>
Edit Calendars button	<p>This button launches the Calendar Editor, which is also accessible from the Unicenter AutoSys JM toolbar.</p> <p>Note • In UNIX, the Calendars button opens the Calendar Definition tool.</p>
Time Zone field	<p>This field allows you to specify a time zone to start jobs. To use the time zone where the job was created, leave this field blank.</p>
Times of Day option	<p>This option enables the text field, in which you type specific times of the day to start the job. Separate each time with a comma.</p> <p>Note • In UNIX, the times of day parameters are accessed directly from the Date/Time Options dialog.</p>



Instructor
Notes

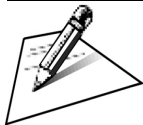
GUI Element	Description
Minutes after Each Hour option	This option enables the text field, in which you type a number that represents a specific interval, such as five minutes after noon, and works with the Run Days group.
Every ____ Minutes	In this field, you can specify the repeat interval for a job; this field also works with the Run Days group so that you can create a job to run every 15 minutes on Fridays.



Interactive Demonstration

Task Purpose: Create a job in UNIX with a date/time dependency. For this example, we will use a date and time representing *now*; that is, the date and time when you are actually performing this exercise.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Definition**.
- 2 In the **Job Name** field, type an appropriate name:
`name_u_jobnow`
- 3 For **Job Type**, select **Command**.
- 4 In the **Command** field, type an appropriate command:
`sleep 90`
- 5 In the **Execute on Machine** field, type your *machinename* or select it from the list.



Instructor
Notes

- 6 In the **Starting Parameters** group, select **Yes** next to **Is the Start Date/Time Dependent** parameter.
- 7 Click **Date/Time Options** to open the Date/Time Options dialog.
 - a Select *today*, where *today* represents the day of the week when you are performing this task.
 - b In the **Times of Day** field, specify a time that is three minutes from now.
 - c Click **Dismiss**. The Date/Time Options are saved when you save the Job Definition.
- 8 Choose **File ▶ Save** and click **Yes** to clear any message boxes. The job is added to the Job Activity Console as INACTIVE.
- 9 To verify, watch the Event Report section of the **Job Activity Console**; the jobnow job should run three minutes later. The Status indicator will update to STARTING, RUNNING, and then SUCCESS.



Interactive Demonstration

Task Purpose: Create a job with a date/time dependency. For this example, we will use a date and time representing *now*; or the date and time you are performing this exercise.

Note • Continue following the naming convention set in the previous task.

- 1 From the Unicenter AutoSys JM toolbar, click **Job Editor**.
- 2 In the **Job Name** field, type an appropriate name:
name_w_jobnow

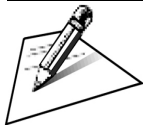


Instructor
Notes

- 3 In the **Command** field, type an appropriate command:
`echo hello`
- 4 In the **Send to machine** field, select your *machinename* from the list.
- 5 Select the **Date/Time** tab.
- 6 Select the **Date/Time Conditions** check box to activate the date/time settings for this job.
- 7 Select *today*, where *today* represents the day of the week when you are performing this task.
- 8 In the **Times of Day** field, specify a time that is three minutes from now.
- 9 Choose **File ► Save** and click **OK** to clear any message boxes. The job is added to the Scheduler Console as INACTIVE.
- 10 To verify, watch the **Scheduler Console**; the jobnow job should run three minutes later. The Status indicator will update to STARTING, RUNNING, and then SUCCESS.

Task Summary

You now know how to schedule specific starting dates or times for your jobs. In the next Task, you will learn how to send alarms to signal job problems such as failures.



Instructor
Notes

Close the Task by reviewing concepts and checking understanding.

Slide 5-6



Task 3: Submit Alarms/Terminators

Alarms

Alarms are informational only; they can be set to inform you if a job fails to execute as intended. For example, to prevent a file from being inadvertently truncated during processing if the job should end too early, you specify a minimum run time (in minutes) for the job. The job should not end in less than the specified time. If the job does end prior to this time, an alarm is generated to alert you to investigate the situation and take corrective action.

Similarly, you can set alarms to notify you if:

- the job's maximum run time was passed
- the job was terminated
- the job failed
- the box containing the job failed

Note • A monitor, the Alarm Sentry, or the Alarm Manager must already be running.



Slide Show slide 6 to introduce the Task.
Define Terms

Instructor
Notes

Terminators

If a job in a box fails, you can select to terminate the entire box, meaning all of the jobs. Or, if the box fails, you can terminate the job.

You can set a time limit in number of minutes after start to terminate the job after starting it.

Example

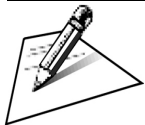
Assume you receive a high level request to run a special report — one time only. The report is on payroll data, and can only be accurate if it is run after the payroll box has run successfully. Since your CEO requested the report, you wish to be notified if the report cannot be run. To do so, you should select the **If this box fails, terminate this job** and the **Send Alarm if this job fails** options.



Interactive Demonstration

Task Purpose: Notify the operator when a job in UNIX fails to run as scheduled. Create a new job (*name_u_testalarms*) that will start a command.

- 1 Open the Job Definition tool and create a new job called:
name_u_testalarms
- 2 In the **Command** field, type:
`sleep 90`
- 3 In the **Execute on Machine** field, type your *machinename*.
- 4 Click **Adv Features**. The Advanced Features dialog opens.
 - a Leave the **Minimum Run Time** at 0.



Instructor
Notes

- b** Set the **Maximum Run Time** at 1 minute.
 - c** Select **Yes** next to **Send ALARM if this job fails?**
 - d** Click **Save&Dismiss**.
- 5 Save and then start your *name_u_testalarms* job.
- 6 Watch the Event Report section of the Job Activity Console for status updates. The job is defined to run a command for 90 seconds, but to terminate if it goes past one minute. The Alarm indicator at the bottom right corner of the Job Activity Console will change color to signal an alarm was received.



Interactive Demonstration

Task Purpose: Notify the operator when a job fails to run. Create a new job (*name_w_testalarms*) that will start the command prompt when run.

- 1 Click **Job Editor**.
- 2 In the **Job Name** field, type an appropriate name: *name_w_testalarms*
- 3 In the **Command** field, type an appropriate command. For this example, use: `cmd.exe`
- 4 In the **Send to Machine** field, select your *machinename* from the list.
- 5 Click the **Alarms/Terminators** tab.
 - a** Select the **Send Alarm if this job fails** option.
 - b** Set the **Maximum Run Time** to 1 minute, but leave the **Minimum Run Time** at 0.
- 6 Choose **File ► Save**.



Instructor
Notes

- 7 Click **OK** to clear the message box. The new file is added to the Scheduler Console.
- 8 Right-click the new job and choose **Start Job**.
- 9 Click **Yes**.
- 10 To verify, check the Status column on the Scheduler Console, which updates from INACTIVE to RUNNING. The command prompt opens in a new window, verifying that the job executed. Leave it open for longer than one minute.

After one minute (which is the time we specified), the Alarm manager should show the MAXRUNALARM type.

- 11 Examine the AutoSys log and note the alarm event.

Task Summary

You now know how to set jobs to send alarms to signal processing problems. In the next Task, you will learn how to use the miscellaneous settings found on the **Misc.** tab of the Job Editor.



Instructor
Notes

Close the Task by reviewing concepts and checking understanding.

Slide 5-7



Task 4: Specify Additional Job Attributes

The **Misc** tab on the **Job Editor** dialog box contains additional attributes. To access these same attributes in UNIX, click the **Adv. Features** button on the Job Definition tool:

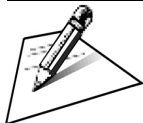
Name	Description
Number of times to restart this job after a failure	<p>This attribute specifies how many times, if any, the job should be restarted after exiting with a FAILURE status. The default is "0", which means the job will not be automatically restarted after an application failure.</p> <p>Note • Setting this attribute will apply only to application or command failures, not to system or equipment failures (such as an unavailable machine).</p>
Time Zone	<p>This attribute contains the alias of the time zone that controls the job's start. For example, when you specify a start time of 6 AM, you can use the Time Zone attribute to specify Eastern, Central, and so on, to coordinate remote jobs.</p> <p>Note • In Windows, this attribute is specified on the Date/Time tab.</p>
Delete Job after completion	<p>This attribute indicates whether or not the job definition should be automatically deleted after successfully executing. You can specify a number of hours: 0 indicates <i>immediately</i>; and a negative value "turns off" the attribute.</p> <p>If the job did not complete successfully, Unicenter AutoSys JM will keep the job definition for seven days before automatically deleting it. This attribute is useful for letting Unicenter AutoSys JM schedule and run a one-time batch job.</p>



Instructor
Notes

Slide Show slide 7 to introduce the Task.

Name	Description
AutoHold on for jobs in boxes	<p>This feature is only for jobs in a box. When a job is in a box, it inherits the box's starting conditions. This means that when a box goes into the RUNNING state, the box job will start all the jobs within it (unless other conditions are not satisfied).</p> <p>For example, you might want to place a job in a box, but not start the job until a non-job (for example, operating system level) event arrives. Selecting the AutoHold on for jobs in boxes option automatically changes the job state to ON_HOLD when the box that contains the job begins RUNNING. At this point, the job is in exactly the same state as if it were manually placed on hold.</p>
Permissions group	<p>By default, only the owner has edit and execute permissions on a job, and all edit and execute permissions are valid only on the machine on which the job was defined. However, the owner can grant different types of permissions when defining a job. This provides users with Edit and Execute permissions on a per job basis.</p> <p>Execute: The user or the world can issue events that affect the running of the job as in STARTJOB, KILLJOB, and so forth.</p> <p>Edit Definition: The user or the world can edit the job definition itself, including deleting the job.</p>



Instructor
Notes

Discuss Group permissions are for UNIX only.

Name	Description
Group:	Users assigned to the job owner's primary group can <i>execute</i> or <i>edit</i> the job (or both) if logged onto the machine where the job was created (the machine specified in the owner attribute, that is, <i>user@machine</i>).
World:	Users can <i>execute</i> or <i>edit</i> the job (or both) if logged onto the machine where the job was created (the machine specified in the owner attribute, that is, <i>user@machine</i>).
All Hosts:	Users, regardless of the machine logged onto, can <i>execute</i> or <i>edit</i> the job (or both) (otherwise, the user must be logged onto the machine specified in the owner attribute, that is, <i>user@machine</i>).

Examples

Number of Times to Restart This Job After a Failure

Consider a job that dials a fax machine to send a specific file, but the line is busy. The file is stored in a cache and will be deleted upon the job's termination - whether it is success or failure. Since you would prefer to avoid having to repeatedly scan the file into the cache, you could use this parameter to specify the number of times to redial the fax machine.



Instructor
Notes

Review Examples

Time Zone

Consider a job that must be run for offices located in New York, San Francisco, and London. If the event processor is located in London, but you want the specified jobs to run using New York time, you would specify Eastern Time Zone in this field.

Delete Job After Completion

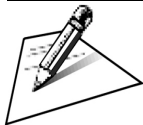
As a Manager, you frequently need to train new Operators to create job definitions. To conserve database space, you want to delete these training files. You could use this parameter to specify to delete the job so many hours after it successfully completes.

AutoHold on for Jobs in Boxes

Use this parameter for operator intervention. Earlier in this course, we used a paycheck printing example. Since printing checks requires check stock loaded in the printer instead of greenbar paper, you may wish to have an Operator confirm that the printer actually contains the required medium before running the job. So, if there are several printing jobs contained in a box, using this parameter automatically changes the job state to ON_HOLD.

Permissions Group

By default, the owner of a job is the user who defines that job on a particular machine. When a user defines a job on UNIX, the user ID is retrieved from the UNIX environment and attached to the job in the form of `user@machine`. The owner is defined by the owner job attribute. By default, only the owner can edit and execute the job.



Instructor
Notes

Using the same paychecks example used previously, consider the Payroll Department. Although the jobs that control paycheck printing were probably defined by IT Operators, they must be run by end users in that department. Therefore, execute permissions must be granted to that group.



Skill Practice

Task Purpose: Practice using miscellaneous settings. (Remember to use the standard naming convention of: *name_[u|w]_job name*)

- 1 Create a box that contains three jobs:
 - a `job_a` is to notify Technical Operations that `job_b` must start. `job_a` has no starting conditions. (Use a simple `echo` command to simulate notification.)
 - b `job_b` depends on the success of `job_a`. `job_b` must be on auto hold until released by the Technical Operations team. `job_b`'s permissions are currently set so that no one but the owner can run it.
 - c `job_c` depends on the success of `job_b`.

Discuss

Did all three jobs start?

If `job_a` failed, what would happen?



Instructor
Notes

Skill Builder: Schedule Jobs



Business Problem

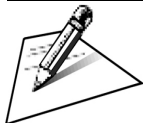
RBC's internal training team wishes to push courseware to students registered for a specific training course. An outside job will execute the command to deliver materials. After the command is complete, the file `c:\temp\reg.txt` will exist.

You must define the jobstream that accomplishes the following:

- Job_1 watches for `c:\temp\reg.txt`, which contains the contact information for each registered student. Job_1 only runs on Mondays.
- Job_2 depends on Job_1 to start. Job_2 simulates the delivery of files to each name on `c:\temp\reg.txt`. Use any simple command here to force a failure status. You wish to be notified if Job_2 fails. Job_2 should take no longer than 10 minutes to execute. If it fails, it should automatically restart once.
- Job_3 simulates a report print-out of Job_2's failure only. Use any simple command. It does not run if Job_2 succeeds.
- Job_4 starts when Job_2 completes successfully. It simulates the printing of completion certificates, which require blank certificates to be loaded in the printer. Therefore, the job should not start until an operator confirms the presence of the proper medium. Use any simple command.

Hint

To simulate delivering materials, use any simple command that forces a FAILURE status, to verify your job. Use all information presented to you so far in this course to create your solution.



Instructor
Notes

Solution Students must create jobs 1, 2, 3, and 4.

Job 1 parameters: Every Monday. File Watcher (`reg.txt` is assumed to have been created by some job outside the scope of this problem.)

Job_2 depends on `s(job_1)`. Simple command. Max run time is 10 minutes. Number of tries to restart is 1. Alarm if failure.

Job_3 depends on `f(job_2)`. Otherwise, it is not run. Simple command.

Job_4 depends on `s(job_2)`. Autohold is on, which means this must be in a box. Simple command.

Slide 5-8



Module Summary

You should now be able to:

- Create and Test Jobs Dependent on Job Status
- Submit a Date/Time Schedule
- Submit Alarms/Terminators
- Specify Additional Job Attributes

In this Module, you learned how to start jobs based on dates or times and on the status of trigger jobs. You also learned how to define jobs to send alarms or automatically terminate. In the next Module, you will learn to use calendars, which extends your job scheduling capability.



Close Module Review the Module Objectives.

Instructor
Notes

Use Calendars

Slide 6-1



Module Objectives

After this module, you will be able to:

- Create Calendars
- Modify Calendars
- Import and Export Calendars
- Schedule Jobs Using Calendars

Module Overview

Most large businesses operate on a global basis. Such operations subject the business to time zone differences, which you learned about in the module called *Schedule Jobs*, and to schedule differences, such as local holidays and customs. In this module, you will learn how to use the Calendar feature to plan your jobs around multiple shifts and holidays, or other schedule variables.



Slide Use slide 1 to introduce the Module Objectives.
Discuss the Overview.

Instructor
Notes

Slide 6-2



Task 1: Create Calendars

Using the Unicenter AutoSys JM Calendar Editor, you can define and maintain calendars that are saved to the database and can then be applied to any job definition. To access the Calendar Editor, click **Calendar Editor** in Windows, or **Calendars** in UNIX, from the Unicenter AutoSys JM toolbar.

The Calendar Editor displays six months of the currently active calendar. The dates on the calendar can be in one of the following states:

Unset

The date is not set. This is the default for any new calendar.

Set

The date is set for this calendar.

Block

The date is not eligible for setting when applying a rule, indicating a conflict.

A *rule* could include such settings as: *the first day of each month*, or *every Thursday*. Conflicts occur when you mark dates, such as all U.S. holidays, as *blocked*. If you tried to apply a rule to run a payroll job every Thursday, you will find that Thanksgiving Thursday is already blocked as a U.S. holiday.

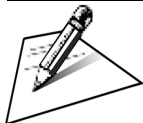
Calendars can be set up as dates to run *or* can exclude the running of certain jobs.

Interactive Demonstration



Task Purpose: Create a holiday calendar to be used for job exclusion in UNIX.

- 1 From the Unicenter AutoSys JM toolbar, click **Calendars**. The Calendar Definition tool opens.
- 2 Choose **File ► New**.



Instructor
Notes

Slide Use slide 2 to introduce the Task.
Define terms.

Discuss This might be a good time to suggest possible naming conventions for calendars. Making generic file names is easiest to keep current.

■ Use Calendars

Interactive Demonstration

- 3 Name your calendar as follows:

name_usholidaysxx

where *name* represents your name and *xx* represents the last two digits of the current year.

- 4 Click **OK**.

- 5 Select the following U.S. holidays for the current year:

January 1

May 26

July 4

September 1

November 27

December 25

- 6 Choose **File ▶ Save**.

- 7 Choose **File ▶ Exit** to close the Calendar Definition tool.

- 8 Click **OK** to confirm the exit.

Your U.S. holidays calendar is now ready to be used to schedule a job. You will test this calendar during *Task 4: Schedule Jobs Using Calendars* later in this Module.

Interactive Demonstration



Task Purpose: Create a holiday calendar to be used for job exclusion in Windows.

- 1 From the Unicenter AutoSys JM toolbar, click **Calendar Editor**. The Calendar Editor tool opens.



Instructor
Notes

- 2 Select the following U.S. holidays for the current year:

January 1

May 26

July 4

September 1

November 27

December 25

- 3 Choose **File ▶ Save As**.
- 4 Name your calendar as follows:

name_usholidaysxx

where *name* represents your name and *xx* represents the last two digits of the current year.

- 5 Select the appropriate Instance and click **OK**.

Note • This calendar will be tested in a later task.



Skill Practice

Task Purpose: Practice creating simple calendars.

- 1 Open the **Calendar Editor** or **Calendar Definition** tool.
- 2 Create a new calendar that includes the last Friday of every quarter.
- 3 Save your calendar as *name_quarterend*.

Task Summary

You now know how to use the Calendar Editor to create a simple calendar. In subsequent Tasks, you will learn how to modify saved calendars, saving valuable time, and how to set up rules, simplifying recurring date selections.



Instructor
Notes

Skill Practice

Ideally, you want your students to manually select the last Friday of every quarter, so you can use this manual hardship as the basis for the Applying Rules topic.

Slide 6-3



Task 2: Modify Calendars

Managing hundreds or even thousands of jobs requires you to work as effectively as possible. Using the Calendar Editor or Calendar Definition tool, you can modify previously created and saved calendars and save them under new names.



Skill Practice

Task Purpose: Modify an existing calendar. In this exercise, you will update the holiday calendar you created in a previous demonstration to reflect the next year.

- 1 From the **Calendar Editor** or **Calendar Definition** tool, open your *name_usholidaysxx* calendar.
- 2 Change the number of years you are viewing.

In UNIX, this can be done by choosing **Options ▶ Date Range ▶ Thru Next Year**.

or

In Windows, this can be done by choosing **Preferences ▶ Years ▶ Two**.

- 3 For each of the following U.S. holidays, select the date it falls on for next year:

New Year's Day

Memorial Day

Independence Day

Labor Day

Thanksgiving

Christmas

- 4 Change the calendar name by saving it as:

name_usholidaysxx

where *xx* represents the last two digits of the next year.

The calendar is then stored and can be used to schedule a job for next year.



Slide Use slide 3 to introduce the Task.

Instructor
Notes

Apply Rules to Calendars

Calendar *rules* simplify date selections; instead of manually selecting all the Thursdays in a year, you can set up a rule that specifies *every Thursday*. You create and apply rules using the Term Calendar Rule tool, accessed from the Calendar Editor's or Definition's **Edit** menu.

Term Calendar Rule

Action

- Set Dates
- Unset Dates
- Block Dates

Date Range

- All in Year: 2003
- All in Range:
 - Starting: 05/09/2003 (MM/DD/YYYY)
 - Ending: 12/31/2003 (MM/DD/YYYY)

Date Selection Rule

Occurrences	Day	Period
<input type="checkbox"/> First	<input checked="" type="radio"/> Day (Any)	<input checked="" type="radio"/> No Period
<input type="checkbox"/> Second	<input checked="" type="radio"/> Weekday	<input type="radio"/> Monthly
<input type="checkbox"/> Third	<input checked="" type="radio"/> Specific Day(s)	<input type="radio"/> Quarterly
<input type="checkbox"/> Fourth	<input type="checkbox"/> Monday	<input checked="" type="radio"/> Every 1 Weeks
<input type="checkbox"/> 1st th	<input type="checkbox"/> Tuesday	<input checked="" type="radio"/> Calendar
<input type="checkbox"/> Next To Last	<input type="checkbox"/> Wednesday	
<input type="checkbox"/> Last	<input type="checkbox"/> Thursday	
<input type="checkbox"/> Every	<input type="checkbox"/> Friday	
<input type="checkbox"/> Every 1 th	<input type="checkbox"/> Saturday	
	<input type="checkbox"/> Sunday	

☐ Rescheduling Rule - for rescheduling conflict dates (for use with Set Dates action only)

Move Direction

- To Previous
- To Following

To Day

- Any Day
- Weekday
- Calendar-based
 - ☐ In Calendar
 - ☒ Not In Calendar

Calendar

OK Apply Cancel



Instructor
Notes

■ Use Calendars

Examples

The screen pictured is of the UNIX version of the Term Calendar Rule tool. The upper portion of the screen is referred to as the **Rule Specification** area, and the bottom portion is called the **Apply Rescheduling Rule** area.

Within the Rule Specification area, you can select the action you wish the rule to perform, the date range to include in the rule, and the rule's occurrences. By combining selections, you direct the rule to be applied to various combinations of dates.

Examples

- If you run a payroll job each week, you might want to set every Thursday of every month for the current calendar. You could select **Every** under the **Occurrences** group, **Thursday** under the **Specific Days** group, and **Monthly** under the **Period** group.
- To block each holiday and prevent those dates from being scheduled, you could use your holiday calendar to make the following selections:

Block Dates (from the Action group)

Every (from the Occurrences group)

Day (from the Day group)

Calendar (from the Period group)

Note • In the Calendar field, you will need to type the name of your holiday calendar or you can click the Calendar button and select your holiday calendar from the list.



Instructor
Notes

Examples Discuss the examples either as a class or in small groups.



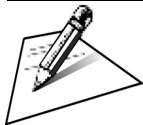
Skill Practice

Task Purpose: Apply rules to calendars on UNIX.

- 1 Open the **Calendar Definition** tool.
- 2 Create a new calendar that includes every Thursday of every month for the current year.
- 3 Save your calendar as: *name_payroll_calendar*
- 4 Define calendar rules with the following criteria:
 - a **Action:** Block Dates
 - b **Date Range:** All in Range
 - c **Occurrences:** Every
 - d **Day:** Thursday
 - e **Period:** Calendar (choose *name_usholidaysxx*)
 - f Define the Rescheduling rule for Thursdays that fall on a holiday
 - g **Move direction:** To Previous
 - h **To Day:** Weekday
- 5 Click **Apply** and then **OK**.
- 6 Choose **File ▶ Save**.

Task Summary

You now know how to modify existing calendars to save time, and how to create and apply rules, to simplify selecting recurring dates.



Instructor
Notes

Slide 6-4



Task 3: Import and Export Calendars

Using the Calendar Editor or Definition tool, you can import calendars from, and export them to text files, making it easy to schedule jobs around shifting demands. Calendars contained in ASCII text files can be saved directly into the database.

- To import a text file containing a calendar, choose **File ▶ Import** from the Calendar Editor. The Open dialog box is displayed, from which you may browse for and select the desired file. In UNIX, the Calendar Definition's **File ▶ Import** option opens an Import File Name dialog from which you may browse and select the desired file.
- To export all calendars currently defined in the database to an ASCII text file, choose **File ▶ Export All**. The Open dialog box is displayed, from which you may select the desired location and file name to which the calendar definition should be saved. It should be noted that exporting all calendars saves them to a single ASCII text file.

In UNIX, the Import and Export dialogs are similar to those in Windows, with the following distinction:

The **Export All** option is not available. Instead, you may choose **Export** from the **File** menu.

Skill Practice



Task Purpose: Export all calendars to an ASCII text file.

- 1 Open the **Calendar Editor** or **Calendar Definition** tool.
- 2 Choose **File ▶ Export**.
- 3 For **Selection**, add the name of the text file, `myCal.s`, to the end of the string.
- 4 Click **OK**.



Instructor
Notes

- | | |
|----------------|--|
| Slide | Use slide 4 to introduce the Task. |
| Discuss | It is important to discuss the lack of filtering when exporting calendars. |



Skill Practice

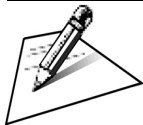
Task Purpose: Import a text file containing calendars.

Calendar names are stored in the database. You can not import a calendar that has the same name as an existing calendar. If you attempt to import a calendar that already exists, the calendar will be skipped. The calendar must be deleted first.

- 1 Open the **Calendar Editor** or **Calendar Definition** tool.
- 2 Choose **File ► Open**.
- 3 Select the text file used previously in the export exercise.
- 4 Using the UNIX *v i* Editor, add the following lines to the end of the file:

```
calendar: name_firstofmonth_year1  
01/01/2003  
02/01/2003  
03/01/2003  
04/01/2003  
05/01/2003  
06/01/2003  
07/01/2003  
08/01/2003  
09/01/2003  
10/01/2003  
11/01/2003  
12/01/2003
```

- 5 Save and close the file.
- 6 From the **Calendar Editor**, choose **File ► Import**.



Instructor
Notes

■ Use Calendars

Skill Practice

- 7 Select the file you previously edited in the UNIX `vi` Editor.
- 8 Click **OK**.

Note • Exporting all calendars saves them in a single ASCII text file.

Task Summary

In this task you learned to import and export calendars that were already created. In the next task you will learn to use those calendars to schedule jobs.



Instructor
Notes

Slide 6-5



Task 4: Schedule Jobs Using Calendars

Once you have created your calendars, you can use them to schedule jobs to never run (*exclusion* calendars) or run (*run* calendars) on the dates specified.

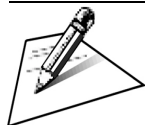
For example, in a previous Task, you practiced creating a `usholidays` calendar. Such a calendar is useful for scheduling job exclusions.



Interactive Demonstration

Task Purpose: Schedule an *exclusion* job in UNIX. In this demonstration, you will practice creating a job to use an exclude calendar created during a previous demonstration.

- 1 Create a new Unicenter AutoSys JM command job named `name_u_testcal`.
- 2 In the **Description** field, type an appropriate description, such as:
Calendar jobs that will not run on date of test.
- 3 In the **Command** field, type: `sleep 90`
- 4 In the **Starting Parameters** area, select **Yes** stating that the Start Date/Time is Dependent.
- 5 Click **Date/Time Options**. A Date/Time dialog is opened.
- 6 Select **Every Day**.
- 7 In the **Do Not Run on Days in Calendar** field, type the percent sign (%) and click **Search** to list all available calendars.
- 8 Double-click your `name_usholidaysxx` calendar to select it.
- 9 Click **Calendars**.



Instructor
Notes

Slide Use slide 5 to introduce the Task.

- 10 Save the file and click **OK**. The job should be successfully added to the database.
- 11 Start the job and check the Unicenter AutoSys JM log. You should see a comment that the job's date/time condition has not been met and the job cannot be started.



Interactive Demonstration

Task Purpose: Schedule an *exclusion* job in Windows. In this demonstration, you will practice creating a job to use an exclude calendar created during a previous demonstration.

- 1 Create a new Unicenter AutoSys JM command job named *name_w_testcal*.
- 2 In the **Description** field, type an appropriate description, such as:
Calendar jobs that will not run on date of test.
- 3 In the **Command** field, type:

`cmd.exe`
- 4 Click the **Date/Time** tab.
- 5 Select the **Date/Time Conditions** option.
- 6 Select the **Run Calendar** option and choose your `usholidaysxx` calendar from the list.
- 7 Save the file and click **OK**. The job should be successfully added to the database.
- 8 Start the job and check the Unicenter AutoSys JM log. You should see a comment that the job's date/time condition has not been met and the job cannot be started.



Instructor
Notes



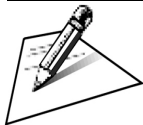
Skill Practice

Task Purpose: Practice creating *run* calendars. In this exercise, you will use the Calendar Editor or the Calendar Definition tool to create a date range in which a job will run.

- 1 Modify your `testcal` job created in the previous demonstration as follows: change the **Date/Time** options to run the following calendar you created in a previous exercise: *name_quarterend*.
- 2 Save and run the job.
- 3 Report your results.

Task Summary

You now know how to schedule jobs using run calendars and exclusion calendars.



Instructor
Notes

Skill Builder: Use Calendars



Business Problem

Due to higher than normal volume during the tax season, your manager at RBC has directed that no jobs be scheduled for April 15th. You want to ensure that your current holiday calendar is modified to add Tax Day as a valid company holiday.

Hint

Use any existing calendar created during prior demonstrations to simulate RBC's "current holiday calendar."



Review invite the students to suggest reasons other than holidays for creating calendars.

Solution Modify a calendar to include April 15th.

Instructor
Notes

Slide 6-6



Module Summary

You should now be able to:

- Create Calendars
- Modify Calendars
- Import and Export Calendars
- Schedule Jobs Using Calendars

In this module you learned to create and modify calendars. Those calendars could be imported and exported for different instances of Unicenter AutoSys JM. Finally, you were able to use the created calendars to schedule jobs.

In the next module, you will create job monitors and browsers, which will help you to manage your jobs more effectively.



Instructor
Notes

Close the Module. Review Objectives and check understanding.

■ Use Calendars

Module Summary



Instructor
Notes



Create Job Monitors and Browsers

Slide 7-1



Module Objectives

After this module, you will be able to:

- Create a Job Monitor
- Create a Browser Report

Module Overview

To effectively manage jobs, you will be asked to provide usage reports or other progress updates to your managers. In this Module, you will learn how to monitor specific types of jobs while they are running and how to set up reports for historical activities.



Instructor
Notes

-
- | | |
|----------------|--|
| Prepare | Before starting this module, you need to set up the PATH for all users on all machines (UNIX side). So that it sets up all user IDs for the students machines. |
| Slide | Use slide 1 to introduce the Module Objectives.
Discuss the Overview. |

Slide 7-2



Task 1: Create a Job Monitor

Monitors are tools that help you retrieve snapshots of the Unicenter AutoSys JM database at your discretion. Monitors work in real time to provide you with a system snapshot. Monitors provide a real-time view of the system. These are the two steps necessary to use a monitor:

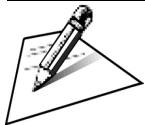
- 1 Define the events to monitor.
- 2 Run the monitor you defined.

A running monitor is an application that polls the database for new events that meet the selection criteria. Monitors are strictly informational. They provide an up-to-the-minute window to events as they occur. For box jobs, you can set a monitor to track all job levels.

Note • Monitors provide a picture of the system's state in real time. If the Event Processor is down, monitors will not provide any information.

Examples

- You just created a new job that is part of a rather complicated jobstream. Although you have tested the job's logic, you decide to monitor this job's execution, just to be certain it will run properly.
- A specific box job has not run successfully, despite repeated troubleshooting efforts. The box contains a total of 30 jobs, but several of those jobs are themselves boxes. You can monitor this box job's execution to track the statuses of all the jobs the box contains, including those contained in the sub-boxes, to track down the problem.



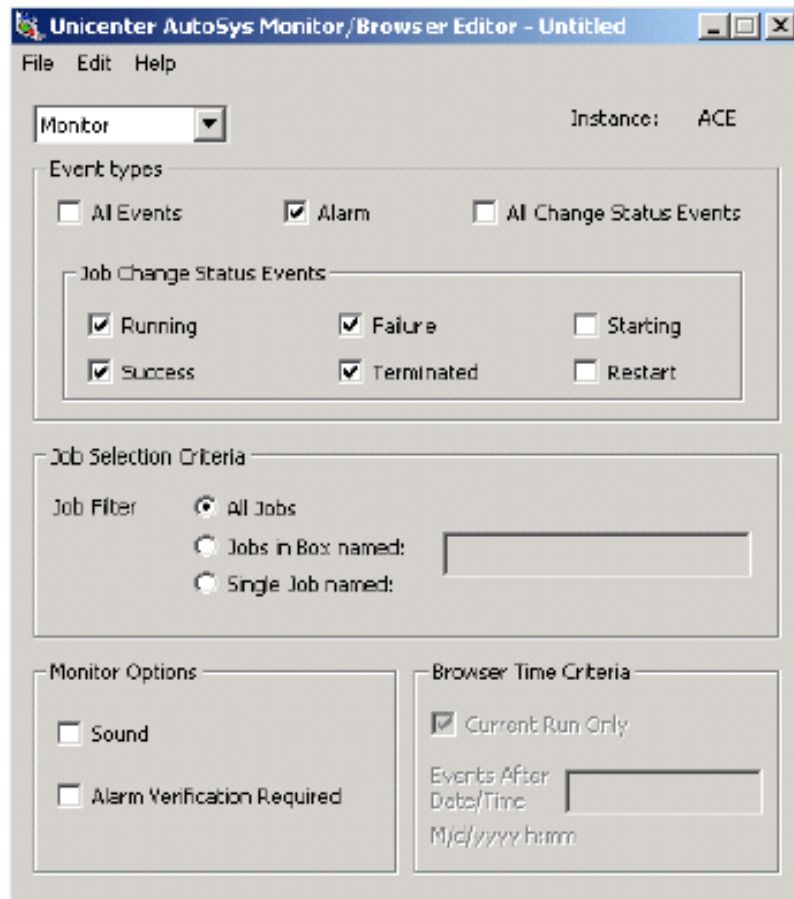
Instructor
Notes

Slide Use the slide to introduce the Task.
 Define terms.
 Discuss the examples.

■ Create Job Monitors and Browsers

Examples

Both Monitors and Browsers are defined using the Monitor/Browser Editor accessed from the Unicenter AutoSys JM toolbar. In Windows, as soon as you select an instance in the Monitor/Browser Editor, Unicenter AutoSys JM establishes a connection to that instance's database until you close the editor. Therefore, using monitors has a high impact on system performance.



The screenshot shows the 'Unicenter AutoSys Monitor/Browser Editor - Untitled' window. It has a menu bar with 'File', 'Edit', and 'Help'. Below the menu bar, there is a dropdown menu set to 'Monitor' and a label 'Instance: ACE'. The main area is divided into several sections: 'Event types' with checkboxes for 'All Events' (unchecked), 'Alarm' (checked), and 'All Change Status Events' (unchecked); 'Job Change Status Events' with checkboxes for 'Running' (checked), 'Failure' (checked), 'Starting' (unchecked), 'Success' (checked), 'Terminated' (checked), and 'Restart' (unchecked); 'Job Selection Criteria' with radio buttons for 'All Jobs' (selected), 'Jobs in Box named:' (with an empty text field), and 'Single Job named:' (with an empty text field); 'Monitor Options' with checkboxes for 'Sound' (unchecked) and 'Alarm Verification Required' (unchecked); and 'Browser Time Criteria' with a checked checkbox for 'Current Run Only' and a text field for 'Events After Date/Time' with the format 'M/d/yyyy h:mm'.



Instructor
Notes



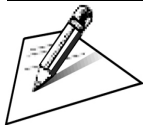
Interactive Demonstration

Task Purpose: Create a Monitor in UNIX.

- 1 From the Unicenter AutoSys JM toolbar, click **Monitor/Browser**.

The screenshot shows the 'Monitor/Browser' dialog box with the following sections and options:

- Buttons:** Clear, Delete, Save, Run MonBro, Exit.
- Name:** A text input field.
- Mode:** Radio buttons for ☒ Monitor and ☐ Browser.
- Search:** A button below the Name field.
- Monitor/Browse these Types of Events:**
 - ☒ ALL EVENTS
 - ☐ Alarms
 - ☐ ALL Job CHANGE-STATUS Events
 - Job CHANGE_STATUS Events:**
 - ☐ Running
 - ☐ Success
 - ☐ Failure
 - ☐ Terminated
 - ☐ Starting
 - ☐ ReStart
- Job Selection Criteria:**
 - Job Filter:** Radio buttons for ☒ ALL Jobs, ☐ Box with its Jobs, and ☐ Single Job.
 - Job Name:** A text input field.
- Monitor Options:**
 - ☐ Sound
 - ☐ Verification Required for Alarms
- Browser Time Criteria:**
 - Current Run Only:** Radio buttons for ☒ Yes and ☐ No.
 - or -**
 - Events After Date/Time:** A text input field.
 - Format:** (MM/DD/YYYY hh:mm)



Instructor
Notes

■ Create Job Monitors and Browsers

Interactive Demonstration

- 2 In the **Name** field, type an appropriate name for the monitor:
`name_u_monstatus`
- 3 Ensure that **Monitor** is selected as the Mode.
- 4 Under the **Monitor/Browse These Types of Events** group, ensure that **ALL EVENTS** has been selected and then select **Alarms**.
- 5 Under the **Job CHANGE_STATUS Events** group, select **ALL Job CHANGE_STATUS Events**.
- 6 Under the **Job Selection Criteria** group, select **Box with its Jobs** for the **Job Filter** and enter a Job Name to be run.
- 7 Click **Save**. The monitor definition is saved to the database.

Note • There is no indication that the monitor has been saved. After a few moments, continue with the following steps.

- 8 From the **Monitor/Browser Editor**, click **Run MonBro**. The Monitor/Browser dialog opens to monitor the indicated job.
- 9 To verify, run any job and watch the window, which will show the events for the job selected.

Note • Remember to manually close the Monitor/Browser window. A new one is created for each monitor you define. Monitors impact system performance.



Instructor
Notes



Interactive Demonstration

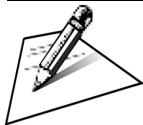
Task Purpose: Create a Monitor in Windows.

- 1 From the Unicenter AutoSys JM toolbar, click **Monitor/Browser Editor**.
- 2 Select **Monitor** from the list.

Note • In the **Event Types** and **Job Change Status Events** groups, all options are selected by default.

- 3 Under the **Event Types** group, ensure that only **Alarm** is selected.
- 4 Under the **Job Change Status Events** group, ensure that only **Running** is selected.
- 5 Under the **Job Selection Criteria** group, ensure that **All Jobs** is selected.
- 6 Choose **File ► Save As**.
- 7 Verify the instance is correct and type an appropriate name for the new Monitor, such as: *name_w_monstatus*
- 8 Click **OK**. The monitor definition is saved to the database.
- 9 From the **Monitor/Browser Editor**, choose **File ► Run Monitor/Browser**.
The Monitor/Browser window opens to monitor the indicated job.
- 10 To verify, run any job and watch the window, which will show the events for that job.

Note • Remember to manually close the Monitor/Browser window. A new one is created for each monitor you define.



Instructor
Notes

Task Summary

You now know what monitors are and how to create them. In the next Task, you will practice creating browser reports, which are similar to monitors, except report on historical events.



Instructor
Notes

Close the Task: Review concepts and check understanding.

Slide 7-3



Task 2: Create a Browser Report

Browsers are reports on the historical activities for a time period you set and are informational only.

You define browsers to display events based on the time the events occurred. Archived events are inaccessible because browsers display only the events still in the database.

You could create a browser that displays:

- the finish times of all database backup jobs run in the current quarter
- all jobs that have an alarm associated with them
- all job levels in box jobs

Browser definitions may be stored in the database so you can run a defined report at any time without redefining the browser.

Examples

- Consider a utility service failure, such as a telephone or power disruption that may have contributed to a set of job failures. Before wasting hours investigating why each job failed, you could run a browser for the period when the outage occurred, to see if all of the failed jobs show up for the same time period.
- Consider a job that ran at 7 AM yesterday. At 7 AM today, you run a daily browser for the last 24 hours. If yesterday's job had failed, it would be listed on the daily browser.



Instructor
Notes

Slide Use the slide to introduce the Task.
Define term.
Discuss examples.



Interactive Demonstration

Task Purpose: Create a Browser in UNIX.

- 1 Click Monitor/Browser.
- 2 Type an appropriate browser name:
`name_u_brower1`
- 3 For **Mode**, select **Browser**.
- 4 Clear the **ALL EVENTS** option.
- 5 Ensure that only **Failure** and **Terminated** have been selected in the Job **CHANGE_STATUS** Events area.
- 6 Under the **Job Selection Criteria** group, select **Box with its Jobs** for the **Job Filter** and enter a Job Name to be run.
- 7 Select **All Jobs**.
- 8 In the **Browser Time Criteria** area, specify the **Events After Date/Time**.
- 9 Click **Save**.
- 10 Click **Run Mon/Bro**.
- 11 Discuss the results.



Interactive Demonstration

Task Purpose: Create a Browser in Windows for a specific date, time, and event status.

- 1 From the **Monitor/Browser Editor**, choose **File ▶ New**.



Instructor
Notes

- 2 Enter an Instance and type a name for the new Browser, such as:
name_w_browser1
- 3 Click OK.
- 4 Select **Browser** from the list.
- 5 For the Event Type, ensure that only **Alarm** is selected.
- 6 For the Job Change Status Events, ensure that only **Failure** and **Terminated** are selected.
- 7 Select **All Jobs**.
- 8 For the Browse Time Criteria, in the **Events After Date/Time** field, enter a specific date/time: *yesterday* at noon, so that you get a report of all failed or terminated jobs since yesterday afternoon. An example would be: 10/28/2003 07:00
- 9 Choose **File ► Save As**.
- 10 Run the Browser to search the database for all failed and terminated jobs since yesterday afternoon.

Browser reports are more commonly used than monitors, since they provide you with a historical snapshot of database events. Browsers connect to the database only to get the specified information, so their impact on system performance is typically far less significant than that of monitors.

Task Summary

You now know what browsers are, how they differ from monitors, and how to create them.



Instructor
Notes

Close the task. Review concepts and check understanding.

Skill Builder: Discuss Monitors and Browsers



Business Problem

A critical piece of RBC's networking equipment has been malfunctioning and frequently goes off-line. The resulting backlog of jobs waiting to be restarted is growing, so the Data Center Manager has asked you to monitor this backlog and report back if any of those jobs currently marked for RESTART returns a FAILURE status when executed.

Hint

Use the CAPITALIZED words to solve this problem.



Solution You could run a monitor on all jobs marked RESTART. However, since monitors have a high impact on performance, you could also consider a browser.

Instructor
Notes

Slide 7-4

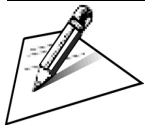


Module Summary

You should now be able to:

- Create a Job Monitor
- Create a Browser Report

In this Module, you learned the distinction between these terms and practiced creating both, as well as the considerations that drive which method of reporting to choose under certain circumstances.



Instructor
Notes

Close the Module. Review Objectives and check understanding.

■ Create Job Monitors and Browsers

Module Summary



Instructor
Notes

Use the Web Interface

Slide 8-1



Module Objectives

After this module, you will be able to:

- Set Up and Customize the Web Interface
- Manage Jobs
- Generate Remote Access Reports
- Run Simulations Using Xpert

Module Overview

The realities of working in a large corporation such as RBC, Inc. mean that you may be required to attend certain meetings, and therefore, cannot be in front of your Unicenter AutoSys JM console all of the time. As you have already learned, you can define jobs that notify you when certain statuses or events result. When you are notified of any problems while away from the console, you can still access Unicenter AutoSys JM using a web interface.

In this Module, you will learn how to gain remote access to Unicenter AutoSys JM and use the Web Interface for a variety of job management tasks, including creating, scheduling, and running jobs, as well as generating reports or testing jobs using a simulated run feature.

Note • Instead of following along through Interactive Demonstrations, your Instructor will demonstrate the Web Interface while you take notes.



Instructor
Notes

- Prepare** For this module to run correctly, you must run the `asrcs_config` command in order to add the T2 instructor machine as a host.
- Slide** Use the slide to introduce the Module Objectives
- Explain** that during this Module, you will demonstrate use of the web interface, while students take notes. All steps are not provided in their Workbooks.
- NOTE: The Web Interface was not fully operational during Course Development, which is why there are so few hands-on exercises and many Instructor only demonstrations.
- Please contact Curriculum Development to suggest ways for improving this Module- specifically, for adding Interactive Demonstrations.

Slide 8-2



Task 1: Set Up and Customize the Web Interface

You can remotely access Unicenter AutoSys JM from any web browser using the Unicenter AutoSys JM Web Interface.

Type the following in your browser's address field:

`http://web_server_name:port_number/autosys/login`

where *web_server_name* represents the name of your web server and *port_number* is the number used if not set to 80.

The login dialog is displayed. Type your *username* and *password* and click **Login**.

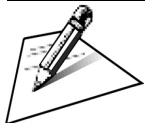
Note • Use the username and password provided by your instructor.

The Web Interface supports two types of login accounts: *Administrator* and *User*.

Administrator The Web Interface Administrator creates or modifies users and adds or deletes instances used by the Web Interface. Web Interface Administrators can also limit a user's access. We recommend you set up a Web Interface Administrator.

User The Web Interface User requires a unique username and password, defined by the Administrator. You can define three types of users: General User, Admin User, and Read Only User

After you log in, the **Web Interface Start Page** is displayed.



Instructor
Notes

Slide Use the slide to introduce the Task.
web_server_name is the instructors t2 machine name
port_number is 80
Username and *password*: autosys autosys
Stress that students must confirm their usernames and passwords with their work site administrators.

■ Use the Web Interface

Task 1: Set Up and Customize the Web Interface

There are two main areas: the **Tree Pane**, which is located down the left edge of the interface and shows the Job Views list, and the **Main Window**, which shows information pertaining to the job objected selected from the Tree Pane. The tabs across the top of the Web Interface provide you with information relevant to the User's profile, set up by the Administrator:

Tab	Options
Job Management	Default view
	The Tree Pane (left edge of the Web Interface) updates according to User privileges. A User may have viewing rights to all jobs in the enterprise, or only to selected views.
	Real Time Monitor
	Job Property Sheet
	Critical Path
	Job Table
	Simulation
Reports	Job
	Forecasting
	Throughput
	Alarm
	Published Reports
	Autosyslog



Name the various parts of the Web Interface.

Instructor
Notes

Tab	Options
Admin	Instance Management
	User Management
	User Mapping
Help	Self-evident

To minimize the load on the Unicenter AutoSys JM database, the Web Interface takes a snapshot of the database's current state and displays the cache. The cache is updated frequently so that the information displayed is current.

Before you can use the Web Interface to monitor Unicenter AutoSys JM, you must configure it to communicate with each instance. This interface can communicate with multiple instances, whereas the Job Editor and some other features previously presented work with a single instance. In UNIX, you are locked into a single instance.



Demonstration

Task Purpose: Add an instance using the **Admin** tab.

- 1 Open your web browser.
- 2 Log in using the default *username* and *password* previously provided by your instructor.
- 3 Click the **Admin** tab and select **Instance Management**. The Instance Management form is displayed.
- 4 To the right of **Choose Action**, select **Create**.



Instructor
Notes

■ Use the Web Interface

Demonstration

- 5 Complete the form with information appropriate for you and your environment:

Attribute	Description
Instance	Name used for each unique instance of Unicenter AutoSys JM; must be 3 letters
Database Type	Oracle, Sybase, or SQL Server
Primary DB Secondary DB	Database hostname and port number
Database Username	Self-evident
Password	Self-evident
Note • Remember to confirm your password.	
RCS or Java Listener Port at EP Host	The server names where the Event Processor resides. The port number used by the EP Log Listener has a default of 4444.
Views	Defines the names of any customized or subset views of the Jobflow. Used if any views were previously defined.

- 6 Click **Submit** to save your configuration settings.

Only your Web Interface Administrator can define views of the enterprise job stream, a feature handy for setting up role-specific access. Permission can be granted for User access to multiple Jobflow Views, JIL, or the Sendevent command.



Instructor
Notes

Step #5	Create the U45 instance using the following information:
Instance	U45
DB Type	Sybase
Primary DB	T2 machine name/ 6324/ autosys
Secondary DB	dcauto2/ 6324/ AUTOSYSDBU45
DB Username	autosys
Password	autosys
RCS/Java Listener Port	4444



Demonstration

Task Purpose: Create a User using the Web Interface.

Watch as your Instructor demonstrates creating users using the Web Interface.

- 1 From the **Admin** tab, select **User Management** in the tree.
- 2 Select **Users**.
- 3 Complete the User Management form and click **Submit**.

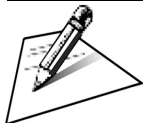


Demonstration

Task Purpose: Create a User Mapping using the Web Interface.

Watch as your Instructor demonstrates creating a User Mapping using the Web Interface.

- 1 From the **Admin** tab, select **AutoSys User Mapping** in the tree.
- 2 Select the appropriate instance.
- 3 Complete the AutoSys Users Information form and click **Verify**.



Instructor
Notes



Demonstration

Task Purpose: Define a view using the Web Interface.

Watch as your Instructor demonstrates defining a view using the Web Interface.

- 1 From the **Admin** tab, select Instance Management.
- 2 Select the appropriate *Instance*.
- 3 On the Instance Management form, in the **Views** field, type the name of a new view.
- 4 Click **Submit**.

Task Summary

You have now seen how to access the Unicenter AutoSys JM Web Interface using a standard web browser. In the next task, you will learn to use the Web Interface to manage and monitor jobs.



Instructor
Notes

Slide 8-3



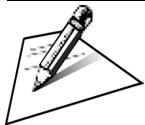
Task 2: Manage Jobs

The **Job Management** tab of the Web Interface provides real-time monitoring of the enterprise or, if your Administrator created views, of specific jobflows that pertain to your role.

When you click the Job Management tab, the Job Views list (left pane) shows each instance currently defined. You can click an instance to expand it. As you do so, the information displayed in the Main Window updates. You can click any object in the Main Window to further expand it.

Directly above the Job Views area, there is a list that contains the following selections:

Real Time Monitor	This is the default view. It provides a map of currently defined box jobs, by default, but you can set it to display all jobs.
Job Property Sheet	This view offers features and functions similar to those of the Job Editor or Job Definition tools covered in earlier modules.
Critical Path	This view provides a graphic representation of the dependencies defined in a jobstream that are critical to the jobstream's eventual success.
Job Table	This view lists certain event parameters for the jobs defined for the selected instance, including the current status, the date and time of the last change status, as well as that of the next scheduled start.
Xpert	This simulation view allows you to test job settings without actually running the job.



Instructor
Notes

Slide Use the slide to introduce the Task.

Note • Again, instead of following along through Interactive Demonstrations, your Instructor will demonstrate the Web Interface while you take notes.



Demonstration

Task Purpose: Use the Real Time Monitor to view all jobs defined for an instance.

- 1 Log onto the Web Interface using the user ID and password supplied by your instructor. After logging on, the **Job Management** tab will be shown by default.
- 2 From the **Job Views** list along the left side of the interface, right-click the instance called **Enterprise**.
- 3 From the shortcut menu, clear the **Box Jobs Only** option, to view defined jobs of all types.
- 4 Right-click **Enterprise** again.
- 5 From the shortcut menu, select **Load Jobs**. You may now expand the Enterprise instance to view the jobs defined within it in one of the following ways:
 - Double-click the instance.
 - Click the icon to the extreme left of the instance.

From the main window, load the job descriptions and then double-click any of the jobs shown to view a description of that job, if one was provided.

From the Job Views list *or* the main window, you may right-click a job to access the same shortcut menu, from which you may start the job, kill the job, change the job's status, or run a job detail report on a specific number of job runs.



Instructor
Notes

To view a job's properties, you may right-click a job object and choose **Edit** from the shortcut menu. The job's Job Property Sheet is displayed. Next to each heading on the Job Property Sheet is an icon that resembles an inverted "V." Click this icon to expand the Property Sheet even further, accessing functions similar to that found on the Scheduler Console or Job Activity Console covered earlier in this course.



Demonstration


Task Purpose: Add a new job using the Web Interface.

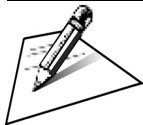
- 1 From the **Job Views** list, click the **Enterprise** instance once to highlight it.
- 2 Directly above the Job Views tree, select **Job Property Sheet** from the list.
- 3 From the toolbar located directly to the right of the list, click **New Command**

Job.



Note • To view the names of each button located on the toolbar, point to each button. A ToolTip will appear, containing the button's name.

- 4 Complete the fields required to define a command job that runs any simple command.
- 5 Click **Save the Current Job.** 
- 6 To confirm your intention to save the job, click **Yes**. The job you created now appears in the Job Views list on the left side of the interface.



Instructor
Notes



Demonstration

Task Purpose: Start a job using the Web Interface.

- 1 From the **Job Views** list, click the **Enterprise** instance once to highlight it.
- 2 Select **Real Time Monitor** from the list.
- 3 Right-click the job you created choose **Start Job** from the shortcut menu.
- 4 Click **Yes** to close the confirmation box.
- 5 Click **OK**.
- 6 Watch the icons next to your job in both the Job View list and the main window for a color change to green, indicating that your job has completed successfully.

Task Summary

In this task you were able to see how to view, add, and start a new job using the Web Interface. In the next task you will learn to use the reporting features of the Web Interface.



Instructor
Notes

Add a brief description of the remaining color changes. Click the color reference button to show what each color represents.

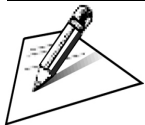
Slide 8-4



Task 3: Generate Remote Access Reports

The Web Interface provides extensive reporting features. The following table describes the types of reports that may be generated from the Reports tab of the Web Interface:

Report Type	Description
Job-Based Report	<p>This report shows specific job-related details, including status distribution and current status.</p> <p>You can generate this report for all instances, all jobs, all views, all machines, all statuses, and specific time periods.</p>
Forecast Report	<p>This shows the jobs scheduled to run over the next 24 hours.</p> <p>You can generate this report for all instances, all views, all machines.</p> <p>You can use this report to make sure your skeleton staff tomorrow can handle the workload currently scheduled.</p>
Throughput Report	<p>This shows the number of jobs processed hourly in an instance.</p>
Alarm Report	<p>This shows all jobs that produced an alarm. You can specify the alarm attributes on which to report.</p> <p>You can generate this report for all instances, all views, all jobs, all machines, all users, and specific dates or times.</p> <p>You can use this report to see what kinds of problems plagued a given day, for trend analysis.</p>



Instructor
Notes

Slide Use the slide to introduce the Task.

Report Type	Description
Job Statistics Report	You can generate this report for all instances, all views, by hour or by application, on a given day. It returns job statistics including the number of events, number of restarts, number of failures, number of force starts and so on.
AutoSys Log	This report provides a real-time view of the Event Processor Log. You can view by instance.



Demonstration

Task Purpose: Explore the Reports tab of the Web Interface. Watch as your Instructor explores the **Reports** tab. Use the space provided at the bottom of this page to take notes.

- To generate a Forecast Report, click the Reports tab and select **Forecast**. Select the instance (your machinename), select the view and set a forecast date. To view all job names, use the % wildcard. Click **Generate Report** to complete the task. The report lists the jobs scheduled to run according to the parameters just set. However, it will not include the dependent jobs.
- To generate a Throughput Report, click the Reports tab and select **Throughput**.
- To generate an Alarm Report, click the Reports tab and select **Alarm**.
- To generate a Job Statistics Report, click the Reports tab and select **Job Statistics**.
- To generate an AutoSys Log report, click the Reports tab and select **AutoSys Log**. This report runs the `autosys log -e` command for remote viewing so you can check a job's status from any computer.



Instructor
Notes



Demonstration

Task Purpose: Create a report using the Web Interface that shows all jobs that have ever run to success.

- 1 Click the **Reports** tab.
- 2 From the Reports list along the left side of the interface, select **Job**. The Job-Based Report Form fills the main window.
- 3 From the Job-Based Report form, in the **Instance** field, select an appropriate instance from the list.
- 4 For **View**, select **All Views** from the list.
- 5 For the following fields, leave all default selections unchanged:

Job Name

Exit Status

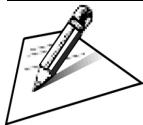
Machine Name

Time Period

- 6 For **Current Status**, select **Success** from the list.
- 7 For **Display Fields**, select all available options.
- 8 Click **Generate Report**.

Task Summary

In this task you learned to generate reports using the Web Interface to provide job status, future job schedules, produced alarms, and a real-time view of the Event Processor Log.



Instructor
Notes

Slide 8-5



Task 4: Run Simulations Using Xpert

Xpert is similar to the timescape function in previous Unicenter AutoSys JM releases (4.0 and earlier). It permits you to simulate jobstreams and job forecasting by letting you select various start or end dates and times.

The Xpert dialog has two tabs: **Configuration** and **Job Selection in View**.

- **Configuration:** Use this tab to access the calendar to specify the Start and End times for the desired job, as well as to view the job's status. The default setting is one hour from the future date/time specified.
- **Job Selection in View:** Use this tab to find a job and select the appropriate machine. You can find jobs by machine name or by job name.

Demonstration



Task Purpose: Simulate a command job.

- 1 Create a command job that will run for a long while and that uses the Date/Time and Minutes After Hour features.
- 2 From the **Job Management** list, select **Xpert**.
- 3 Click **Apply**. This will load all date/time jobs and any jobs that are dependent on the date/time jobs. You should see the job you created in Step 1.

Notice the time range is set to the default of one hour from the date/time. This can be changed by changing the start/end date/time.

- 4 From the toolbar, click **Start Simulation (Play)**. This begins the simulation. You will see a vertical bar moving from left to right that shows how the job is running. Notice that when the vertical bar passes over the minutes after the hour you specified when you created the job, the box changes color to indicate **RUNNING**.



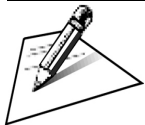
Slide Use the slide to introduce the Task.

Instructor
Notes

When the simulation is complete, the job color changes again to indicate TERMINAL status.

- 5 From the toolbar, click **Stop Simulation**. This ends the job simulation.

Note • While a simulation is running, you cannot switch views unless you first stop the simulation.



Instructor
Notes

Skill Builder: Discussion



Business Problem

You are attending an RBC department head meeting all day. Although you prepared a presentation, your boss asks you for an immediate status update on a series of jobs that are either currently running now, or scheduled for later today. How would you gain immediate access to view the job statuses while in this meeting?



Instructor
Notes

Solution

Use this Skill Builder to enhance the learning, if you feel that the students should need additional help.

During Development, this feature was unavailable and so, is not fully documented yet. Please contact Curriculum Development as you gain experience using this new feature and we will include in the student workbooks.

Students should log onto the Web Interface as previously taught. Select the Enterprise instance, Load Jobs, then select Job Table to view statuses of currently defined jobs for that instance.

Slide 8-6

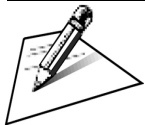


Module Summary

You should now be able to:

- Set Up and Customize the Web Interface
- Manage Jobs
- Generate Remote Access Reports
- Run Simulations Using Xpert

In this module, you learned to use the Web Interface to create and manage jobs, generate reports, and simulate a command job.



Instructor
Notes

Close the Module. Review Objectives and check understanding.

Course Summary

Slide 8-7



You should now be able to:

- Manage Jobs
- Manage Events
- Use the Scheduler or Job Activity Console
- Use Basic Commands
- Schedule Jobs
- Use Calendars
- Create Job Monitors and Browsers
- Use the Web Interface

Throughout this course, you learned and practiced the core components of managing jobs using Unicenter AutoSys JM. For additional practice, consider the following end-of-course Skill Builder.



Instructor
Notes

Slide Use the slide to discuss the Course summary.
Check for understanding before introducing the End of Course Skill Builder

Skill Builder: End of Course Summary



As a class, discuss the various approaches you might take to define the jobstream described. If time permits, decide which portions of your plan to practice during class, right now.

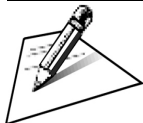
Business Problem

RBC's enterprise operates on the following real-time model: as inventory levels are depleted, purchase requisitions are automatically created to reorder inventory, and routed for the required approvals. Once the approvals have been granted, Purchasing issues a Purchase Order and the order is placed with the appropriate vendor. If approvals are not granted for whatever reason (an Approver is out of the office), that job must notify an Operator and trigger another job that escalates the requisition to alternate Approvers.

When a Purchase Order is issued, a job notifies Accounts Payable and Receiving. Invoices linked to Purchase Orders are paid only when Receiving acknowledges that the materials were delivered by forwarding an electronically signed Delivery Receipt file. Usage reports are generated based on this information at weekly, monthly, quarterly, and annual intervals.

Only when Accounts Payable has a valid Purchase Order and a signed Delivery Receipt file from the Receiving dock, will any invoices be paid. Checks are printed on the 15th and last day of every month, unless either date falls on a weekend or holiday, at which time the checks would be printed on the previous Friday.

Discuss the questions on the following page.



Instructor
Notes

Possible Solutions

One possible solution requires at least 11 jobs, see the Solutions Appendix.

The critical path must include: Low inventory > Purchase Reqs > Approvals > Purchase Orders > Notification > Delivery.

Discuss optimizing this jobstream; there will be more than one way. As for boxes, we put the notification jobs in a single box.

Play "What if" with the class. There are limitless options, but try to explore those the class identifies as "typical".

Questions

- How many jobs do you estimate are required for such a process?
- What is the critical path to such a jobstream?
- What is the optimal way to define all the required dependencies so that the critical path is assured? What jobs could be in a box?
- What information, if any, is missing from this problem that would help you make better job management decisions?



Instructor
Notes

Possible SolutionS.

This is a class discussion; such a complex jobstream will require too much time to actually create during class, so encourage students to discuss the jobs required to handle this process.

Try to draw a jobstream diagram on the white board similar to the one pictured in Module 5. One possible solution is illustrated in the Solutions Appendix in the Student Workbooks.

Alternatively, you could ask students to use flip chart paper or the white board to jot down the job definitions for each of the jobs this problem requires.



Skill Builder Solutions

Skill Builder: Module 1



Business Problem

The Sales Team at Really Big Corporation uses a specific report on a regular basis, a sales report that lists the sales by product for the period specified. This report is called `sales_report.txt` and is generated by running a batch file called `sales_report.bat`. You need to know when the `sales_report.txt` file is available. You know you must create at least two jobs:

Job	Purpose
Use <code>name_[u w]_joba</code> as the job name.	to run the command that produces the report
Use <code>name_[u w]_jobb</code> as the job name.	to watch for the sales report file

How would you create and manage these jobs?

Solution

Create **jobA** to run the command `sales_report.bat`, and **jobB** to watch for a file called `sales_report.txt`. **jobB** should give a status of successful. when the file it is watching becomes available. They should create a third job, a box job that contains **jobA** and **jobB**.

Verification

Files will be found at: `c:\TEMP\`



Notes

Skill Builder: Module 2 Class Discussion



Business Problem

It is 5 PM on a weekday. Every day at the close of business, after RBC has completed its daily transactions, you must run an end-of-day program set. Assume this program set contains a series of jobs named `job1` through `job20`. About halfway through this set of programs, data is written to tape (`job9`). However, today, your colleague informs you that the tape drive is out of tape. `job9` is part of the critical path and cannot be skipped. What should you do?

After `job9` completes, `job10` prints a summary of the day's activities. The printer paper is jammed. Since `job10` is not part of the critical path, it may be skipped. What should you do?

After the printer jam is cleared and `job10` completes successfully, `job11` is supposed to start. However, perhaps due to the paper jam or other network problem, `job10` is showing an incorrect status, preventing `job11` from starting. What should you do?

Discussion Responses

`Job 9` should be placed on hold, since it is a part of the critical path and nothing else should run until the problem is resolved.

Since `job10` is NOT part of the critical path, subsequent jobs can run without it. Place `job10` on ice.

`Job10` should receive a change status to SUCCESS, or `Job11` could be force started.



Notes

Skill Builder: Module 3



Business Problem

RBC's internal auditors have asked you to create and run special jobs to verify accounting records. Since these jobs are not part of your usual work load, you wish to watch them yourself. Initially, you would like to view these jobs based on their status. Then, for jobs that show a status of FAILURE, you wish to view them by their end times.

Hint

Apply what you have learned in this course so far to solve this problem. You must first create the jobs to run. Use the following job names in your solution:

Job Name	Description
name_acctbox03	contains the jobs called name_acct04_03 and name_accountfile
name_acct04_03	runs account.bat
name_accountfile	watches for account.txt
name_acctbox02	contains the jobs called name_finfile and name_accounting02
name_finfile	watches for financial.txt
name_accounting02	runs acct_annual.bat



Notes

Solution

Create and run the jobs listed in the table. Next, you should filter the jobs by the two box job names to view just these jobs. Finally, you should sort the failed jobs by their end times.



Notes

Skill Builder: Module 4



Business Problem

For each of the following bullet points, discuss what command and options you should use to solve the stated problem.

- RBC's production web server received an alert that thresholds are running too high. You need to generate a written report whenever the threshold is reached.
- If the production web server should become unreachable while it is running a job, you want to detect that the machine is down, receive an alarm, and notify the job owner.
- RBC's week-ending job failed to run as scheduled. You must determine why.
- RBC policy requires you to verify and report on license usage.
- One of the jobs affected by the production web server issue was a box job that was just terminated. Before restarting it, you want to examine the logic of all jobs contained in the box, and verify their current statuses.
- You have examined the jobs contained in the box job and verified their statuses. You just restarted the job, but now want to monitor the statuses as the box runs.
- You are short-handed today and another operator just went home sick. To handle the workload, you need to start a job sooner than scheduled and explain why.

Discussion Responses

autorep	chase	chk_auto_up
autoflags	job_depends	autosyslog -e
sendevent Force Start Job with comment		



Notes

Skill Builder: Module 5



Business Problem

RBC's internal training team wishes to push courseware to students registered for a specific training course. An outside job will execute the command to deliver materials. After the command is complete, the file `c:\temp\reg.txt` will exist.

You must define the jobstream that accomplishes the following:

- Job_1 watches for `c:\temp\reg.txt`, which contains the contact information for each registered student. Job_1 only runs on Mondays.
- Job_2 depends on Job_1 to start. Job_2 simulates the delivery of files to each name on `c:\temp\reg.txt`. Use any simple command here to force a failure status. You wish to be notified if Job_2 fails. Job_2 should take no longer than 10 minutes to execute. If it fails, it should automatically restart once.
- Job_3 simulates a report print-out of Job_2's failure only. Use any simple command. It does not run if Job_2 succeeds.
- Job_4 starts when Job_2 completes successfully. It simulates the printing of completion certificates, which require blank certificates to be loaded in the printer. Therefore, the job should not start until an operator confirms the presence of the proper medium. Use any simple command.

Solution

Students must create jobs 1, 2, 3, and 4.

Job_1 parameters: Every Monday. File Watcher (`reg.txt` is assumed to have been created by some job outside the scope of this problem.)



Notes

■ Skill Builder Solutions

Skill Builder: Module 5

Job_2 depends on `s (job_1)`. Simple command. Max run time is 10 minutes. Number of tries to restart is 1. Alarm if failure.

Job_3 depends on `f (job_2)`. Otherwise, it is not run. Simple command.

Job_4 depends on `s (job_2)`. Autohold is on, which means this must be in a box. Simple command.

Discuss your solutions with the rest of the class.



Notes

Skill Builder: Module 6



Business Problem

Due to higher than normal volume during the tax season, your manager at RBC has directed that no new jobs be added to the Monday morning schedule until after April 15th. You want to ensure that none of your Operators schedule jobs for the beginning of any week until after tax season. What could you do to avoid adding to the high workload?

Solution

Modify a calendar to include Mondays until April 15th. Create a command job that uses that calendar as an exclusion calendar. You could also use a NOT RUNNING dependency.



Notes

Skill Builder: Module 7 Discussion



Business Problem

A critical piece of RBC's networking equipment has been malfunctioning and frequently goes off-line. The resulting backlog of jobs waiting to be restarted is growing, so the Data Center Manager has asked you to monitor this backlog and report back if any of those jobs currently marked for RESTART returns a FAILURE status when executed.

Hint

Use the CAPITALIZED words to solve this problem.

Discussion Responses

You could create a monitor on all jobs marked RESTART and FAILURE. However, since monitors have a high impact on performance, you could also consider creating a browser.



Notes

Skill Builder: Module 8 Discussion



Business Problem

You are attending an RBC department head meeting all day. Although you prepared a presentation, your boss asks you for an immediate status update on a series of jobs that are either currently running now, or scheduled for later today.

Hint

You need immediate access to Unicenter AutoSys JM to learn job statuses.

Discussion Responses

- 1 Log onto the Web Interface using the ID and password; you should check with your site Administrator for real passwords, once you return to work.
- 2 Right-click the appropriate instance.
- 3 Load the jobs.
- 4 Select **Job Table** and make your report to your boss.



Notes



Skill Builder: End of Course Discussion

As a class, discuss the various approaches you might take to define the jobstream described. If time permits, decide which portions of your plan to practice during class, right now.

Business Problem

RBC's enterprise operates on the following real-time model: as inventory levels are depleted, purchase requisitions are automatically created to reorder inventory, and routed for the required approvals. Once the approvals have been granted, Purchasing issues a Purchase Order and the order is placed with the appropriate vendor. If approvals are not granted for whatever reason (an Approver is out of the office), that job must notify an Operator and trigger another job that escalates the requisition to alternate Approvers.

When a Purchase Order is issued, a job notifies Accounts Payable and Receiving. Invoices linked to Purchase Orders are paid only when Receiving acknowledges that the materials were delivered by forwarding an electronically signed Delivery Receipt file. Usage reports are generated based on this information at weekly, monthly, quarterly, and annual intervals.

Only when Accounts Payable has a valid Purchase Order and a signed Delivery Receipt file from the Receiving dock, will any invoices be paid. Checks are printed on the 15th and last day of every month, unless either date falls on a weekend or holiday, at which time the checks would be printed on the previous Friday.

Discuss the questions on the following page.



Notes

Discussion Questions

- How many jobs do you estimate are required for such a process?
- What is the critical path to such a jobstream?
- What is the optimal way to define all the required dependencies so that the critical path is assured? What jobs could be in a box?
- What information, if any, is missing from this problem that would help you make better job management decisions?

Discussion Responses

- One possible solution requires at least 11 jobs
- The critical path must include: Low inventory > Purchase Reqs > Approvals > Purchase Orders > Notification > Delivery.
- There is more than one way to optimize this jobstream. As for boxes, put the notification jobs in a single box.



Notes

■ Skill Builder Solutions

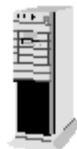
Skill Builder: End of Course Discussion



Notes

Instructor Appendix

Classroom Setup



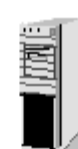
UNIX Server1

Autosys Server v4.5
Instance Name: U45
Autosys Client v4.5
Configured for U45
UNIX Installation CD



UNIX Server2

Autosys Server v4.5
Instance Name: U45
Autosys Client v4.5
Configured for U45
UNIX Installation CD



UNIX Server3

Autosys Client v4.5
Configured for U45
UNIX Installation CD



T1 - Windows

Autosys Client v4.5
Configured for W45
And Configured for U45
ReflectX
Windows Installation CD



T2 - Windows

Autosys Server v4.5
Instance Name: W45
Autosys Client v4.5
and Configured for U45
Configured for W45
Access Control
Instructor Installed Instance: N45
ReflectX
Windows Installation CD
WEB Server



Student Windows

Autosys Client v4.5
Configured for: W45
and Configured for U45
Students will configure for N45
ReflectX
Windows Installation CD

Uninstall Information

To clean up Unicenter AutoSys 4.5 JM, the following instructions were excerpted from the Unicenter AutoSys 4.5 Uninstall Guide document.

[Source - Mike Diaz]

AutoSys 4.5 Win32 - Complete Manual Uninstall Checklist

Note • This should always be completed prior to installing a new AutoSys build. All items should be manually checked and if they exist, deleted.

Remove Unicenter Instance (ACE)

Stop Services: (**Control Panel ▶ Services**)

Stop all of the following services:

Sybase

- ☐ SYBBCK_LOCAL_BS (bundled SYBASE only)
- ☐ SYBMON_LOCAL_MS (bundled SYBASE only)
- ☐ SYBSQL_LOCAL (bundled SYBASE only)

AutoSys

- ☐ CA Unicenter AutoSys JM Event Processor (ACE)
- ☐ CA Unicenter AutoSys JM Remote Agent (ACE)
- ☐ CA Unicenter AutoSys JM Remote Command Server (ACE)

CCI

- ☐ CA-Unicenter (NR-Server)
- ☐ CA-Unicenter (Remote)
- ☐ CA-Unicenter (Transport)

eTrust Access Control

(Stop using **START ▶ Programs ▶ Computer Associates ▶ eTrust ▶ Access Control ▶ Stop eTrust Access Control**)

- ☐ SeOS Agent
- ☐ SeOS Engine
- ☐ SeOS Policy Manager
- ☐ SeOS TD
- ☐ SeOS Watchdog

Registry Stuff (Run regedt32 or regedit)

Open the registry. In HKEY_LOCAL_MACHINE on Local Machine\SOFTWARE\ComputerAssociates\

Delete the following:

- ☐ UnicenterAutoSys
- ☐ eTrustAccessControl
- ☐ CA-Unicenter\SOFTWARE

Delete the following:

- ☐ SYBASE
- ☐ SYBASE_OCS

In HKEY_LOCAL_MACHINE on Local Machine: \SYSTEM\Current Control Set\Services

Delete the following:

- ☐ CCI_NR_SERVER
- ☐ CCI_REMOTE
- ☐ CCI_TRANSPORT
- ☐ CA Unicenter Autosys Event Processor (ACE)
- ☐ CA Unicenter Remote Agent (ACE)
- ☐ CA Unicenter Remote Command Service (ACE)
- ☐ SeOS Agent
- ☐ SeOS Engine
- ☐ SeOS Policy Model (autosys)
- ☐ SeOS TD
- ☐ SeOS Watchdog
- ☐ seosdrv

System Environment

Delete the system variables:

- ☐ CAIGLBL0000
- ☐ CAILOCL0000
- ☐ CLASSPATH
- ☐ DPATH
- ☐ Include
- ☐ InstallPath
- ☐ SYBASE

Delete the following from the PATH PATH:

- ☐ Any path that contains sybase
- ☐ Any path that contains cci

Files and Directories

(Use Explorer) Go to c:\Program Files\CA

Delete the following folders:

- ☐ CCS
- ☐ eTrustAccessControl
- ☐ UncienterAutoSysJM.Common
- ☐ UncienterAutoSysJM.ACE

Program Files

Delete your AutoSys.ACE Folder from your START program menu.

Right-click **START**, explore all users. Look under .../Profiles/All Users/Start Menu/Programs. Right click **AutoSys.ACE**, and delete

Reboot!!!

