

# Using Logical Conditions

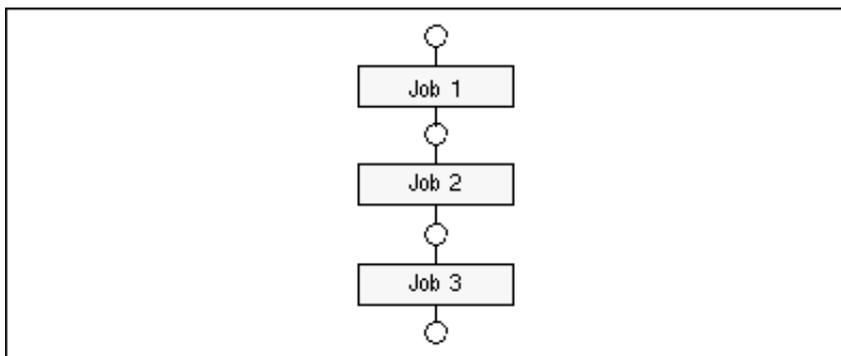
This subsection covers the following topics:

- Shaping a Job Network
  - Integrating Manual Actions
  - Handling Job Failure
  - Combining Input Conditions
  - Not Executing Jobs in Selected Runs
- 

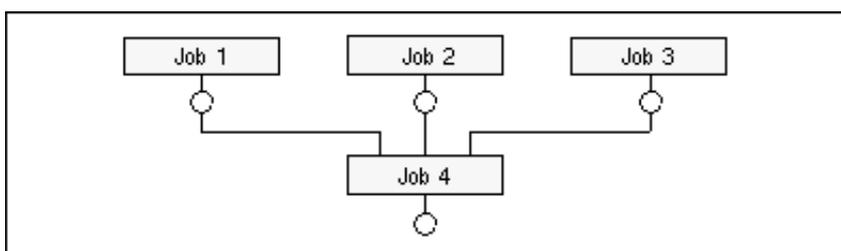
## Shaping a Job Network

Using the job network, job and conditions maintenance facilities, Entire Operations allows you to define job networks that can reflect any set of circumstances or requirements of your batch job processing environment.

For example, logical conditions can be used to link jobs sequentially within a network, as illustrated below:



Logical conditions can also be used to allow parallel processing of jobs, as illustrated by the following figure:



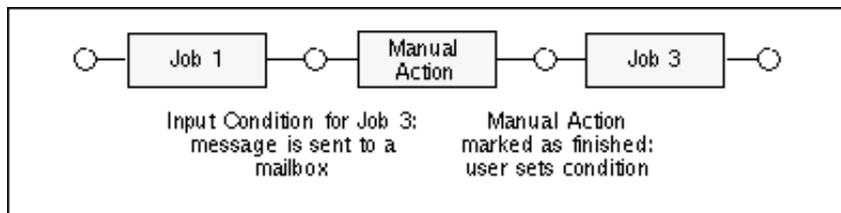
Parallel processing of jobs does not mean that the jobs **must** be executed at the same time. It only means that they are not interdependent and therefore **can** run at the same time.

The occurrence of bottlenecks when parallel jobs compete for system resources can be avoided through the use of resources.

## Integrating Manual Actions

Logical conditions can also be defined for manual actions (for example: loading a tape). You can assign a mailbox name to a condition (for example: Tape loaded OK) (see the subsection Mailboxes in this section). A user associated with the same mailbox is notified if this condition is pending, that is, if the tape is not loaded. The user can then load the tape and manually set the condition to allow Entire Operations to proceed to the next job.

The following figure illustrates this example of manual processing within a job network:

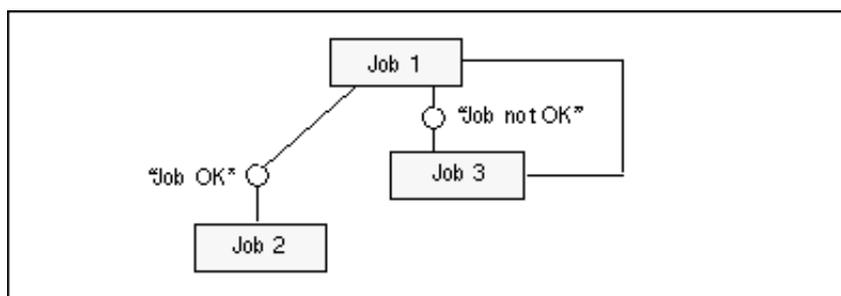


## Handling Job Failure

As mentioned above, you can define several output conditions for a job. Output conditions are set by defined events occurring during job processing and determine how Entire Operations is to proceed after job termination. A job can thus have different subsequent jobs, depending on end-of-job status.

A possible usage of output conditions is to define two possible paths, one selected by a Job OK output condition, the other by a Job not OK output condition. This allows you to define jobs which are only started if a defined error occurs during job processing.

The following figure illustrates this usage of output conditions:



The output condition Job not OK is defined as input condition for Job 3. After Job 3 is finished, Job 1 can be rescheduled.

### Note:

See also the subsection Defining Recovery Action in the section End-Of-Job Checking and Actions.

## Combining Input Conditions

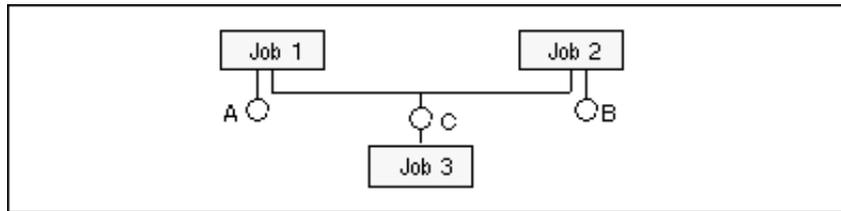
- Example: A or B is TRUE
- Example: Complex Combinations

By default, Entire Operations assumes that all input conditions for a job must be TRUE before it can be submitted. For example, assuming input conditions A, B, C and D are defined for a job, Entire Operations does not submit the job until the Boolean expression A AND B AND C AND D applies.

### Example: A or B is TRUE

However, it is possible to define other logical combinations. For example, if a job is to be submitted if condition A OR B is TRUE, you can define a condition C to be set when either condition A or B is TRUE.

The following figure illustrates this combination:



Condition C is defined as input condition for Job 3 and is set:

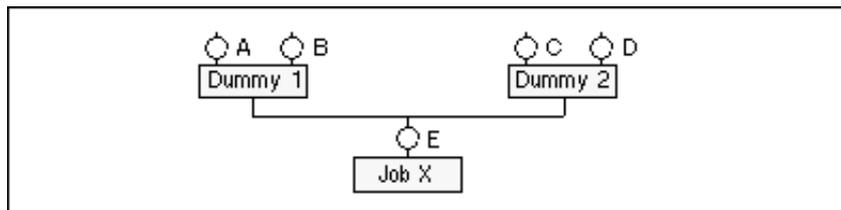
- by Job 1 when condition A is TRUE, or
- by Job 2 when condition B is TRUE.

## Example: Complex Combinations

More complex combinations are also possible. For example, assuming the following Boolean expression must apply to submit a job:  $(A \text{ AND } B) \text{ OR } (C \text{ AND } D)$ , you can define an input condition E for the job which is set when:

- Conditions A and B are both TRUE, or:
- Conditions C and D are both TRUE.

The following figure illustrates this combination of input conditions:



You must define 2 dummy jobs which serve to 'collect' input conditions. Dummy jobs are not submitted to the operating system. They merely wait until conditions A and B or C and D are TRUE, and their execution consists only of setting output condition E as input condition for Job X.

In this way, Entire Operations allows any combination of input conditions.

## Not Executing Jobs in Selected Runs

Entire Operations has the concept of **temporary dummy jobs**. For selected runs and without altering the network structure, a job that is 'real' can be changed into a dummy job that is not started in the operating system.

A job can be changed into a dummy job in the following ways:

1. Schedule dependency (see the subsection Defining Schedule Dependency for a Job in the section Job Maintenance).
2. DUM reference in an input condition. This can be dependent on a symbol value or on the existence of a file (see the subsection Field Descriptions: Master Input Condition Addition / Modification in the section Job Maintenance).