Concepts and Facilities

Version 5.4.3

December 2017

This document applies to Version 5.4.3 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: OGC-ONOPCONCEPTS-543-20180305

Table of Contents

Preface

What Is Entire Operations?	Gives an overview of Entire Operations features to define, control and schedule job networks.
What Is the Entire Operations GUI Client?	Describes the purpose of the Entire Operations GUI Client.
Entire Operations Objects	Explains Entire Operations objects, such as owners and user IDs, networks, logical conditions, mailboxes, resources, calendars, schedules and symbol tables.
Entire Operations Components	Provides information on Entire Operations components, such as the master database, external JCL, active database, operating system, monitor, user interface, logging facility, reporting facility and editing facility.
Entire Operations Facilities	Describes a number of menu-driven Entire Operations facilities which are used to define objects to the system and control and monitor network processing.

I

1 What Is Entire Operations?

Introduction

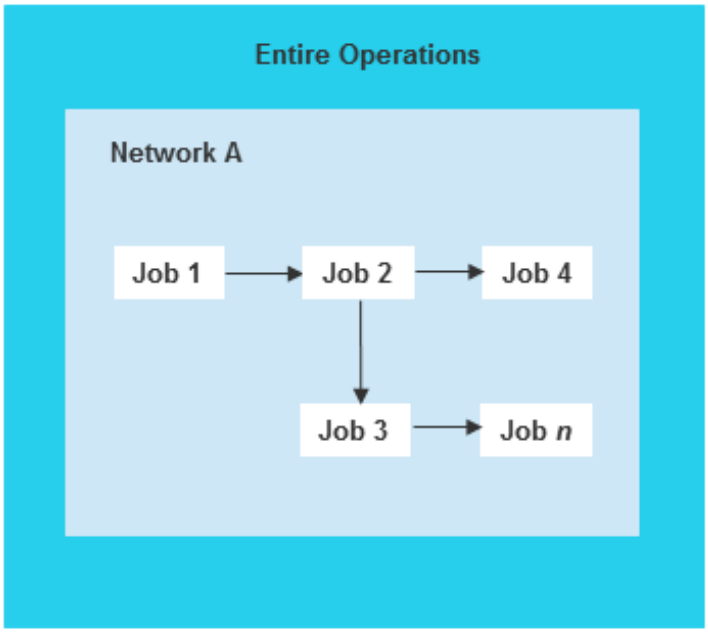
Entire Operations (product code: NOP) is Software AG's system for the automated control and scheduling of job networks. It provides all the functions required to define any type of background processing.

Entire Operations behaves according to the rules specified using a comfortable user interface that guides the user with task-oriented features.

The user interface of Entire Operations is available in English and German. Each user can set the language individually.

Entire Operations requires no modification to underlying operating system or to any of the sub-systems installed at your site. Existing JCL can be put under Entire Operations control unchanged, allowing a smooth transition of your existing production control methods to automatic scheduling.

Entire Operations schedules and controls job networks:



Standard security packages such as RACF, ACF2, CA-TOP SECRET or SECOS are supported, allowing Entire Operations to honor existing security profiles.

For the execution of batch jobs and scripts, Entire Operations uses clearly defined interfaces to installed spooling systems or equivalent operating system utilities.

More than a Scheduler

In addition to the core functionality of Entire Operations described in the previous sections, Entire Operations offers the following features:

- Control Across Operating Systems
- Entire Operations in a Multiple Operating System Environment
- Intelligent Process Control
- Integration of Other Applications
- Integration of People in Automated Operations

Control Across Operating Systems

- **Mainframe environments:**

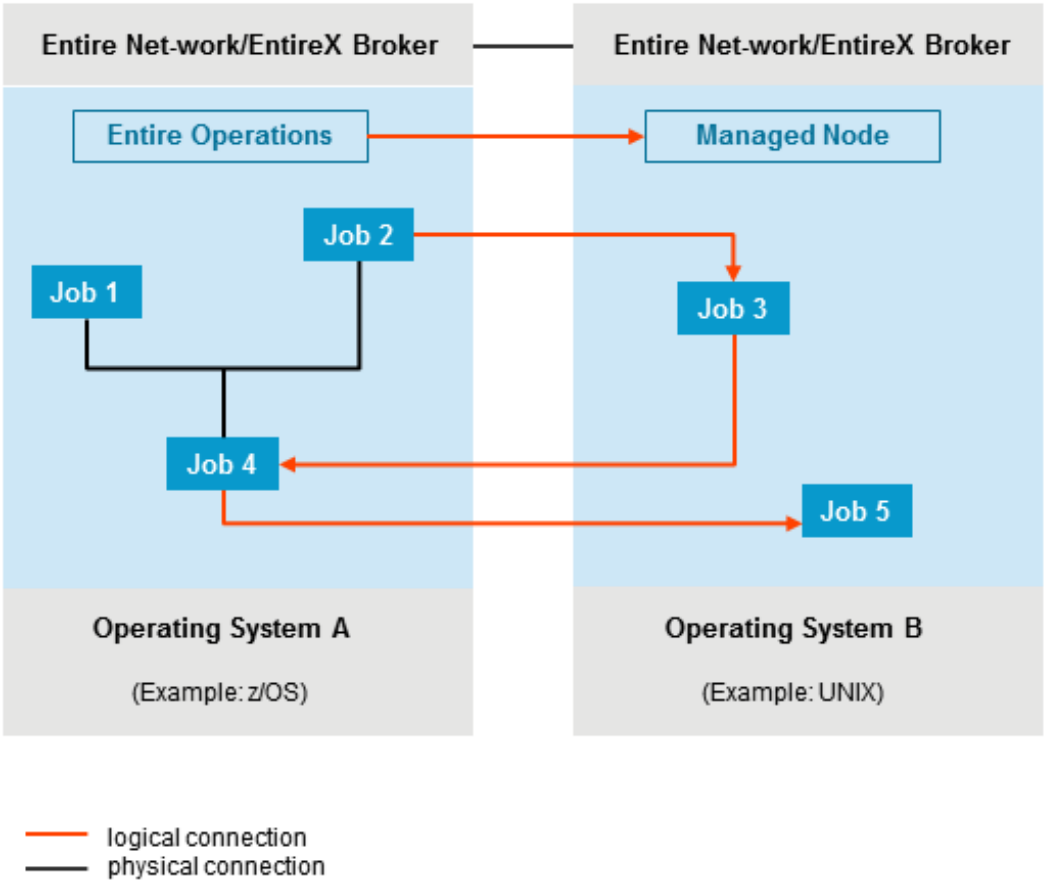
  Entire Operations can be installed on any of the mainframe operating systems z/OS, z/VSE and BS2000 running with TP monitor/online systems, such as Com-plete, CICS, TSO, IMS TM, *open*UTM or TIAM.

- **UNIX:**
  Entire Operations can also be installed on a number of UNIX platforms. It controls production processes on UNIX systems (for example, HP-UX, SINIX RM, AIX, Sun Solaris, Linux), in conjunction with a mainframe or UNIX environment. You can manage your production from any operating system on which Entire Operations can run, or use Entire Operations to control several UNIX and Windows machines.

- **Windows:**
  Entire Operations controls the production processes under Windows.

Background processes running in heterogeneous multi-CPU configurations can also be controlled and monitored by Entire Operations if the computers are interlinked with Software AG's communication service Entire Net-Work. In such a distributed environment, job networks can consist of processing steps that execute on different operating systems.

Entire Operations in a Multiple Operating System Environment



While the Entire Operations Monitor program handles the distribution and decentralized execution of processing steps, this type of distributed processing can still be monitored and controlled centrally from a single point of management.

Intelligent Process Control

Automating any type of operation requires thorough advance planning in order to map the processes in the language of the automation tool. Experience shows, however, that tomorrow's reality often differs from today's plan. One can, of course, anticipate a number of exceptional situations and plan accordingly; and it goes without saying that Entire Operations provides a wide range of features that enable it to react appropriately whenever such a situation arises.

However, the extent to which you can anticipate possible system situations and their consequences is limited; especially in large job network topologies, each new fork in processing adds to the system's complexity, which is neither desirable nor necessarily helpful.

Entire Operations addresses this problem by allowing you to define variable processing steps. For example, executable job control can be built dynamically according to current system conditions such as disk space, content of system queues and availability of specific files. In other words, job control can “read” a current situation and adapt accordingly.

Natural programs and user exits allow you to map any conceivable decision criterion and put all relevant data at your disposal, thanks to Natural interfaces to all commonly used data management and operating systems - even and especially in heterogeneous computer networks.

It is then no longer necessary to think of each and every possible problem situation and spend time and money in defining remedial action before processing starts. All you need to do is specify strategies (= programs) that recognize and rectify problems as they occur.

Integration of Other Applications

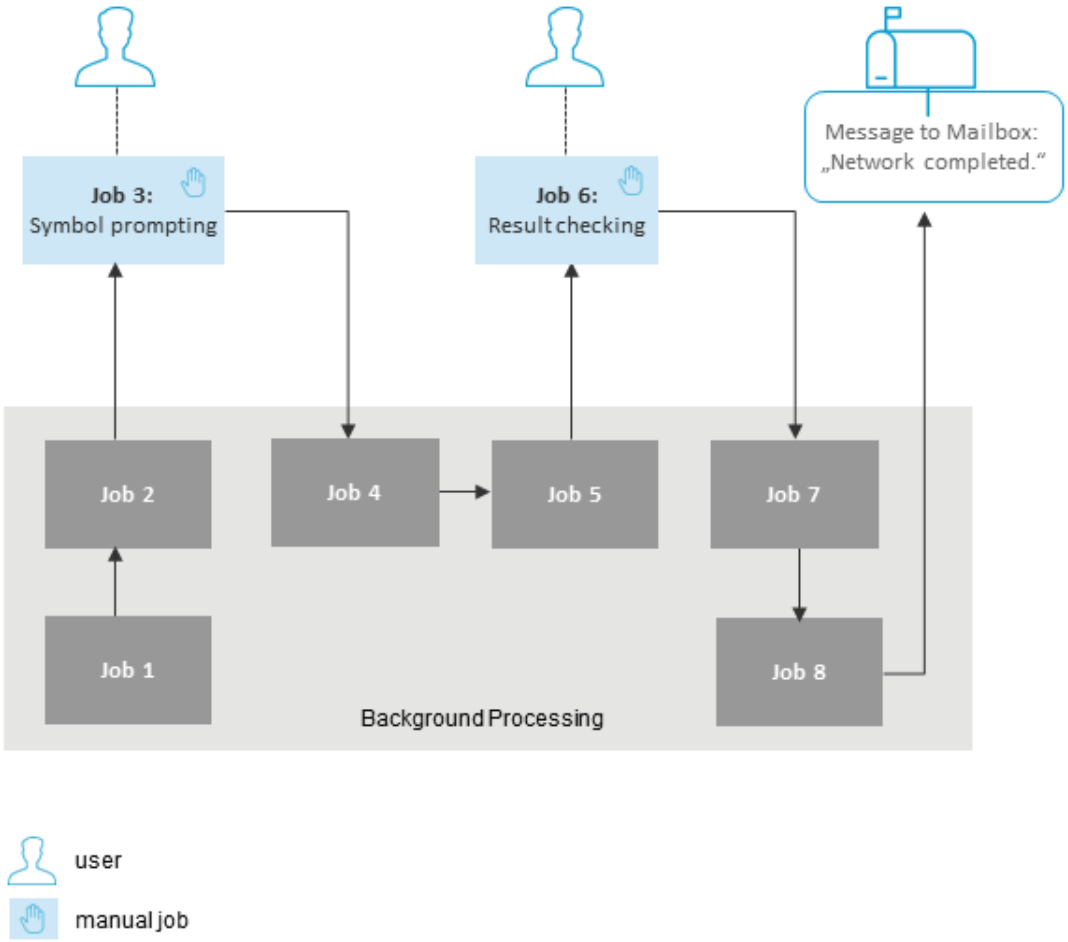
Have you never been annoyed by the sheer number of applications or programs you need for your daily work? Depending on the software configuration at your site, you either have to go through tiresome logon and logoff procedures, or suspend and resume application sessions using a session manager (often to the detriment of system resources).

Entire Operations provides the option to integrate other applications. If any other products from the Entire Systems Management line are installed, Entire Operations recognizes and includes them. These are, for example:

- Entire Output Management
- Entire Event Management
- Entire System Server
- Natural ISPF

Integration of People in Automated Operations

Aside from this technical integration capability, Entire Operations can also integrate people by providing a mailbox concept to combine background processing with user interaction:



At specified times during background processing, messages or prompts can be sent to such mailboxes. This has the twin effect of halting processing and informing all users with access to the mailbox of the situation. These users can react as the situation demands, for example, check a result or enter a symbol when prompted to continue processing.

This mechanism enables selected employees to provide input relevant to their department, while background processing as a whole still remains under central control.

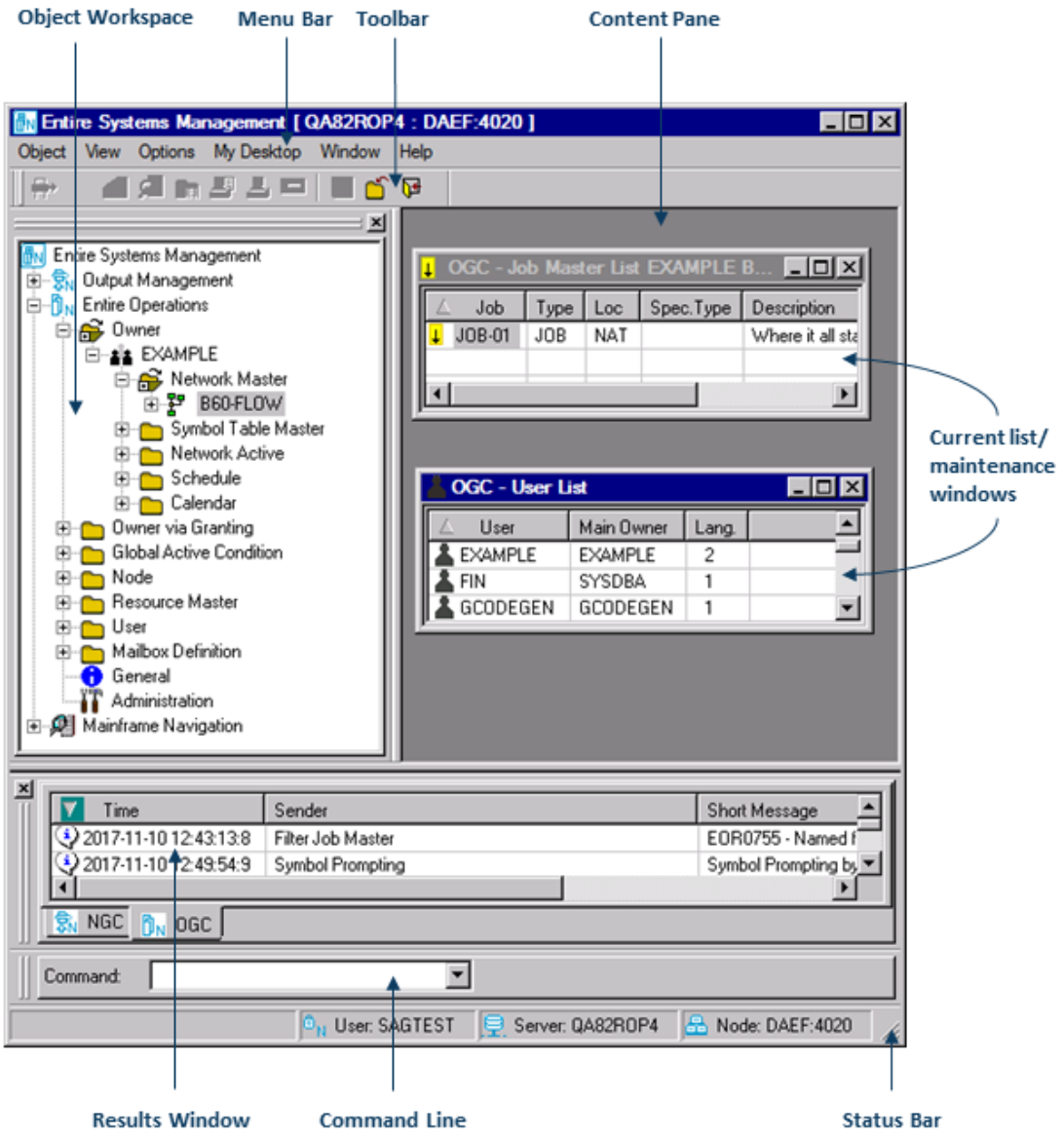
Summary of Benefits

The basic advantages of using Entire Operations to automate your data processing tasks can be summarized as follows:

- Transparent support of several computer nodes, even in heterogeneous environments comprised of z/OS, BS2000, z/VSE, UNIX and Windows operating systems;
- Available in many mainframe and UNIX environments supported by Natural, for example: Com-plete, CICS, TSO, IMS TM, CMS, *open*UTM and TIAM;
- Available in English and German;
- Existing JCL runs unchanged under Entire Operations;
- No modifications are necessary to the operating system;
- Use of dynamically-built JCL or scripts, thus integrating the latest information from the operating system or any available database at execution time;
- Integration of online users into batch network processing through the concept of mailboxes;
- Open interface to user applications: information from Entire Operations can be included in any business application, users can provide input data necessary for daily or future production runs.

2 What Is the Entire Operations GUI Client?

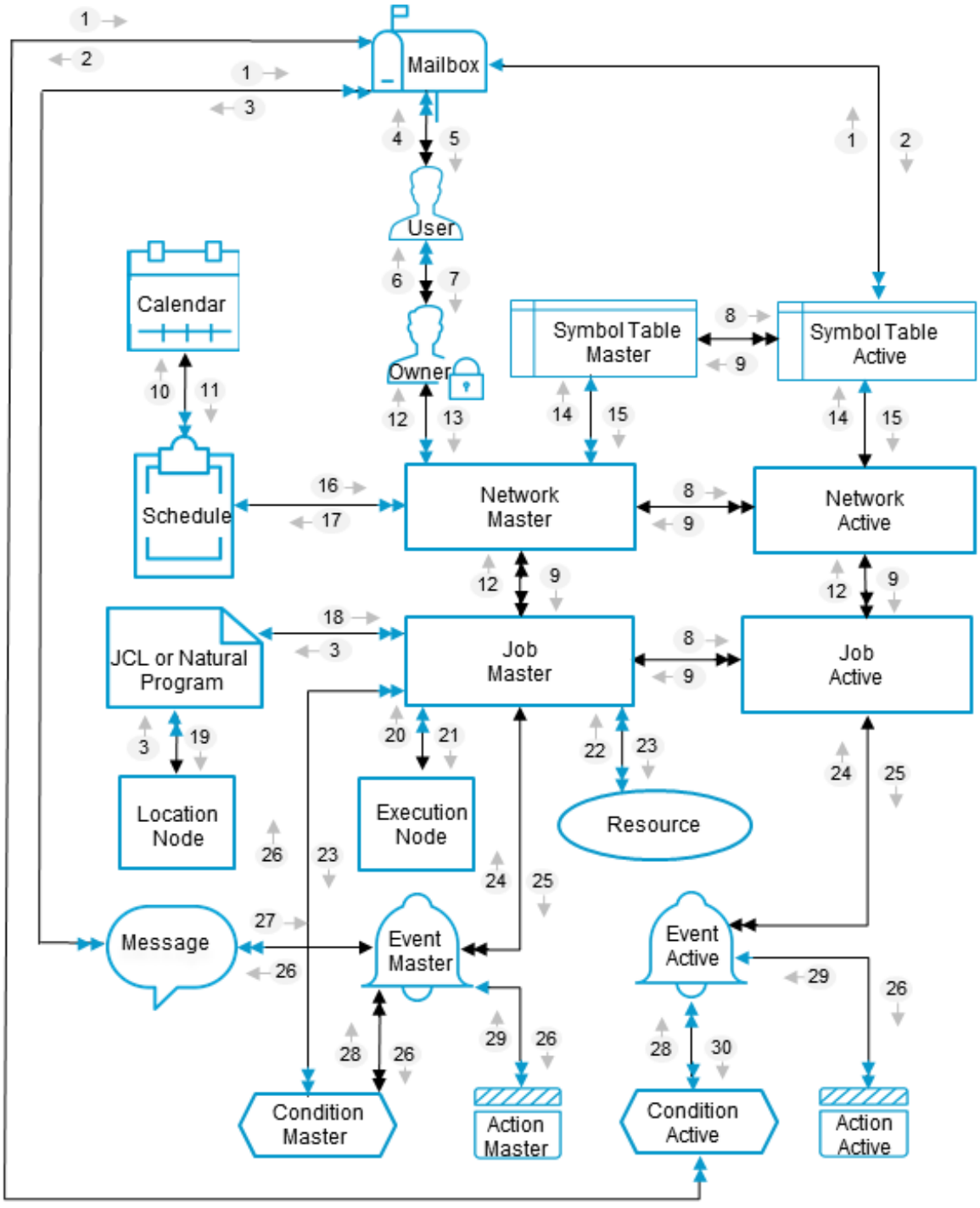
The Entire Operations GUI Client (product code: OGC) is used to perform Entire Operations functions on a mainframe or a UNIX platform from the PC:



For detailed information, see the section *Elements of the Main Application Window* in the *User's Guide*.

II Entire Operations Objects

3 Entire Operations Object Relationship


Mailbox
User
Calendar
Owner
Schedule
Symbol Table Master
Symbol Table Active
Network Master
Network Active
JCL or Natural Program
Job Master
Job Active
Location Node
Execution Node
Resource
Message
Event Master
Event Active
Condition Master
Action Master
Condition Active
Action Active

Legend:

▶	must be only one
▶	none or at most one
▶▶	must be at least one, can be more than one
▶▶	none or one or many
→	direction indicator
1	is sent to
2	contains prompting for
3	contains
4	uses or can use
5	contains messages for
6	authorizes
7	can choose
8	lends properties to
9	is composed of
10	is based on
11	is the basis of
12	belongs to
13	owns
14	obtains parameters from
15	contains values for
16	schedules
17	is scheduled on the basis of
18	is executed as
19	resides on
20	is platform for
21	runs on
22	is preliminary condition for

23	depends on or can depend on
24	determines result of
25	is checked for
26	triggers
27	is set according to
28	is set or reset according to
29	is performed according to
30	sets

4 Owners and User IDs

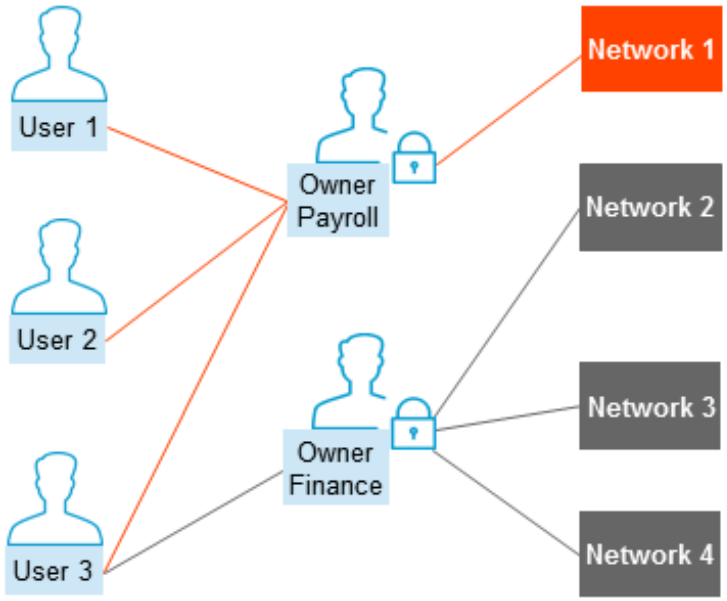
Each Entire Operations user is assigned a user ID for the purpose of security, profiling, message switching and logging.

Each user ID is associated with a user profile which contains authorizations. A user profile can be modified by an authorized user (e.g. the system administrator).

Owner names enhance security and ease of use by grouping user IDs and associating job networks to these group names. A user can be authorized to use several owner names: switching owner names means selecting another group of job networks, which can then be maintained.

Users, Owners and Job Networks

The following figure provides an example of the relationship between users, owners and job networks:



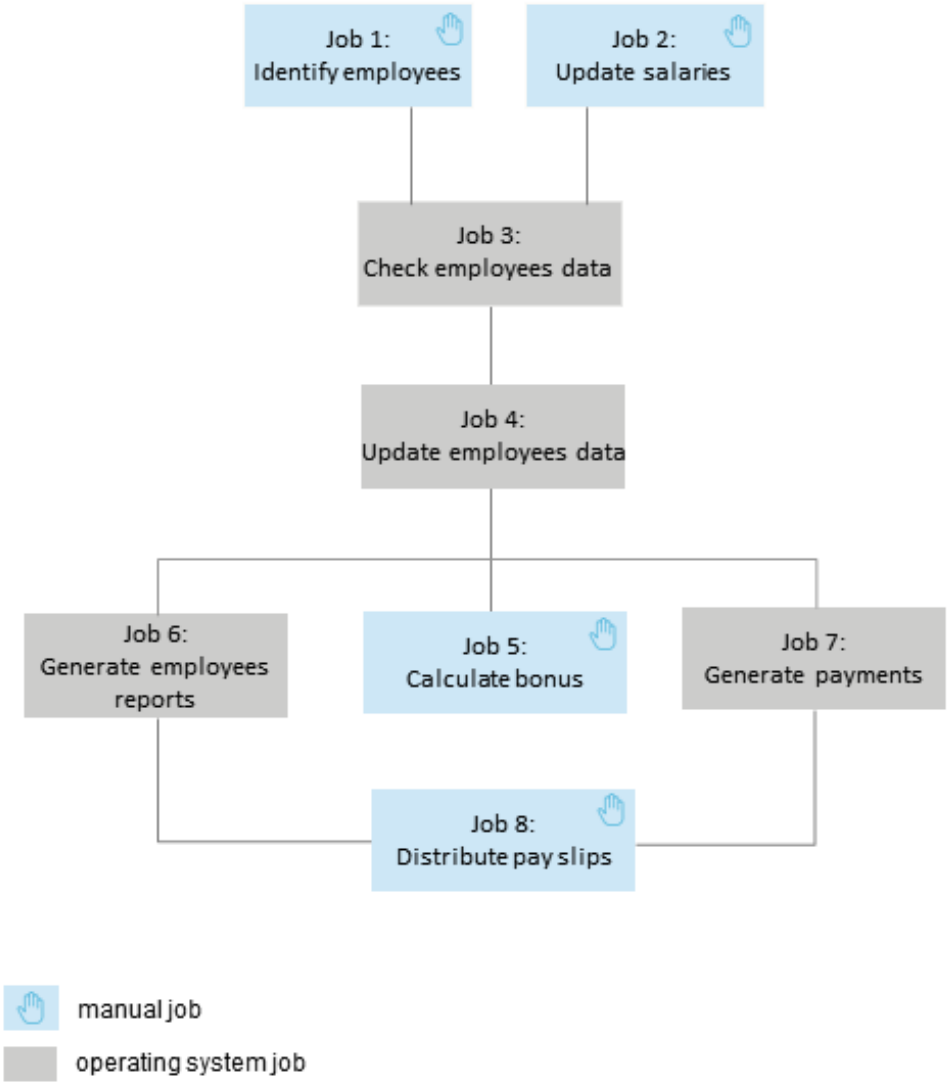
All three users may maintain networks belonging to the owner Payroll, while User 3 is also authorized to access networks belonging to owner Finance.

For more detailed information, see the sections *Entire Operations User IDs* and *Owner* in the *User's Guide*.

5 Job Networks

In Entire Operations, a Job Network is a group of jobs, tasks, scripts or processes that may or may not be interrelated and which are scheduled for work according to a job network schedule table. Thus a job network can represent any unit of business work in production work flow.

You can even integrate into these job networks manual tasks, which are to be performed at fixed times by data center personnel. The following figure illustrates an example of a job network for the automatic generation of pay slips and summary reports as it might be implemented in a payroll department:



In the normal case, a job network consists of a chain of jobs linked together by certain dependencies (e.g. in the simplest case: If Job 1 finishes OK, start Job 2). These dependencies are expressed by logical conditions.

A network is the smallest unit that can be activated automatically by Entire Operations. By using calendars, networks can be automatically scheduled for work by the Entire Operations Monitor. An authorized user can manually activate the networks.

Several active copies (or activations) of a network can work in parallel, since Entire Operations identifies each copy uniquely by its run number, which is automatically assigned to each network at activation time.

For more detailed information, see the following sections:

- ***Versioning of Job Networks***
- *Job Network* and *Network Maintenance* in the *User's Guide*

6 Jobs

The job is one of the basic objects of Entire Operations. Entire Operations regards a job as a user-defined task that is performed by JCL instructions and job IDs, scripts or files as supported by the operating system, Entire Operations subnetworks or dummy jobs, or Natural programs. For all types of jobs supported by Entire Operations, see ***Job Types*** described in the following section.

For detailed information on using and maintaining jobs, see the sections *Job* and *Job Maintenance* in the *User's Guide*.

Job Types

Entire Operations supports the following types of jobs:

- Standard jobs of the operating system (z/OS, z/VSE, BS2000);
- Started Tasks (z/OS);
- Standard shell scripts of the UNIX operating system;
- BAT files on Windows systems;
- Other scripting environments on UNIX and Windows (e.g.: Perl, Windows Scripting Host);
- Command-line oriented executables on UNIX and Windows;
- FTP jobs;
- SAP jobs;
- Natural programs;
- Data file generation;
- Windows services;

For more information about job types, see the section *Available Job Types* in the *User's Guide*.

In addition, for non-CPU jobs there are:

- Dummy jobs to create time windows for non-CPU jobs or Boolean connections for single conditions.

Job Attributes

Each job in the network is defined by a series of identifying attributes:

- (logical) name
- job type
- node ID for job location
- node ID for job execution
- starting time
- end-of-job information
- user ID under which job is run

Such a job can be contained in several job networks.

Jobs in a Multi-Machine Environment

When Entire Operations is used in a Multi-Machine Environment, the location of a job (i.e. the location of its contents) and the location of its execution node can differ: at activation time Entire Operations reads the job information from the source node and executes it on the target node.

Jobs in a network can be interlinked by using ***Logical Conditions***.

See also: ***z/OS: JES2 /*ROUTE Statement***

Job Control Considerations

This section covers the following topics:

- z/OS: JES2 /*ROUTE Statement

- UNIX

z/OS: JES2 /*ROUTE Statement

If a z/OS JES2 JCL contains a statement

```
/* ROUTE XEQ <target>
```

the job will be executed on the target machine.

As long as the SYSOUT will be passed back to the submission machine, the running job is not accessible. Entire Operations detects a rerouting, and behaves differently for such jobs.

For rerouted z/OS jobs, some features do not work, e.g.

- Direct execution tracking.
- Browsing of SYSOUT while the job is executing.
- Cancelling

However, if the SYSOUT is available again, all end-of-job checking and end-of-job actions can be performed.

UNIX

To use the .profile file

Execution of UNIX shell scripts: To allow the usage of profiles in non-login scripts (as they are submitted by Entire Operations), the follow profile handling was added:

1 During submission of a UNIX shell script, the Entire Operations Monitor checks whether the symbol ETC-PROFILE exists in the symbol table of the active job, or in another symbol table in the standard symbol search hierarchy, up to SYSDBA/A

   - If the symbol ETC-PROFILE is found, and if it contains "Y", the batch frame (*.BF) script will source */etc/profile and /etc/profile.local* (only if they exist)
   - The user's script (*.B) will find the environment variables of the profile scripts being set.

2 During submission of a UNIX shell script, the Entire Operations Monitor checks whether the symbol ENV exists in the symbol table of the active job, or in another symbol table in the standard symbol search hierarchy, up to SYSDBA/A.

   - If the symbol ENV is found, and if it is not empty, its content is assumed to be a startup script like $HOME/.profile.
   - If the startup script exists, it's name will be assigned to the environment variables ENV and BASH_ENV by the “batch frame” (*.BF) script.

- If the user's script (*.B) is a Bourne shell, Borne again shell, or Korn shell script, the shell executes the content of the previously set ENV resp. BASH_ENV environment variable.

Note: The user is responsible to make the used profile scripts proof against multiple execution, e.g. by using the PROFILEREAD variable like in Linux.

7 Logical Conditions

For more detailed information, see the sections *Logical Conditions*, *Defining and Managing Job Input Conditions* and *End-of-Job Checking and Actions* in the *User's Guide*.

For information about the prerequisite checking, see the section *Prerequisite Check* in the *User's Guide*.

Logical Condition Overview

The use of logical conditions is the central concept of Entire Operations. Logical conditions are used to describe job or network dependencies. A logical condition can be set by any CPU or manual event. This event must occur before Entire Operations can proceed to the next step.

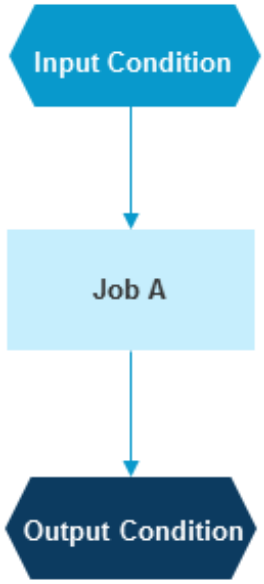
When a job network is activated, each logical condition is assigned a run number. This run number enables Entire Operations to distinguish between the same event that occurs during different network activations.

Logical conditions can be used in two different ways:

- As input conditions;
- As output conditions.

Input and Output Conditions

The following figure illustrates the concept of input and output conditions in relation to a job:

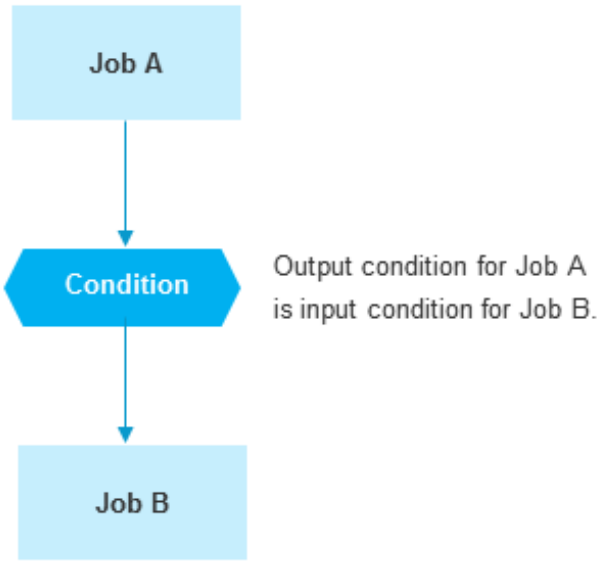


All input conditions must be fulfilled before a job can be submitted (prerequisite condition). You can define any number of input conditions for a job.

An output condition can be set or reset according to the result of predefined events (either automatically given by Entire Operations or user-defined). As part of end-of-job analysis, Entire Operations checks for the occurrence of such events. Several output conditions can be set or reset for each event at the job or even job step level.

Jobs Linked by Input and Output Condition

Jobs in a job network are linked by defining an output condition of one job as an input condition for the next job, as illustrated by the figure below:

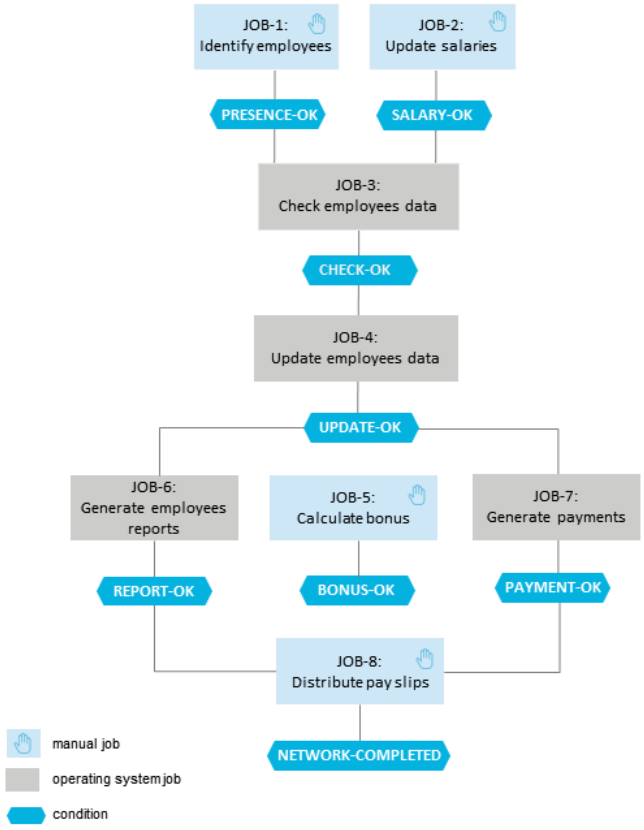


A specified event occurred as a result of Job A. This sets the condition, signaling to Entire Operations that Job B can be started.

You can also link jobs together which belong to different job networks or which are executed on different computer nodes.

Job Network with Logical Conditions

The following figure is an example of the job dependencies for the job network of a payroll department:



The following table gives an overview of the job dependencies (logical conditions) that link the jobs illustrated above:

Job Number	Input Condition	Output Condition
JOB-1	n/a	PRESENCE-OK
JOB-2	n/a	SALARY-OK
JOB-3	PRESENCE-OK	CHECK-OK
	SALARY-OK	
JOB-4	CHECK-OK	UPDATE-OK
JOB-5	n/a	BONUS-OK
JOB-6	UPDATE-OK	REPORT-OK
JOB-7	UPDATE-OK	PAYMENT-OK
JOB-8	REPORT-OK	NETWORK-COMPLETED
	PAYMENT-OK	

For example, Entire Operations will not start JOB-7 (Generate payments) until the input condition UPDATE-OK is fulfilled (this condition is also defined as the output condition for JOB-4).

This job flow is completely independent of the operating system platforms on which the individual processing steps run.

8 Mailboxes, Message Sending

Within Entire Operations, Mailboxes serve to send network-related messages and requests to users and/or groups of users. These messages can be used to inform users about the current status of the job network or to request some data needed for further execution.

Such messages and requests can be:

- User-defined messages.
- System messages from the Entire Operations Monitor.
- Requests for logical conditions, which must be confirmed by any person. These conditions are defined as dependent on any manual action.
- Requests for pending symbol prompting for a scheduled network activation in the near future.
- Using the concept of mailboxes Entire Operations can treat non-CPU-driven tasks in the same way as CPU-driven tasks:
  - tasks can be made dependent on logical conditions and can also set logical conditions;
  - assign a mailbox to these logical conditions to specify who should be informed about them.

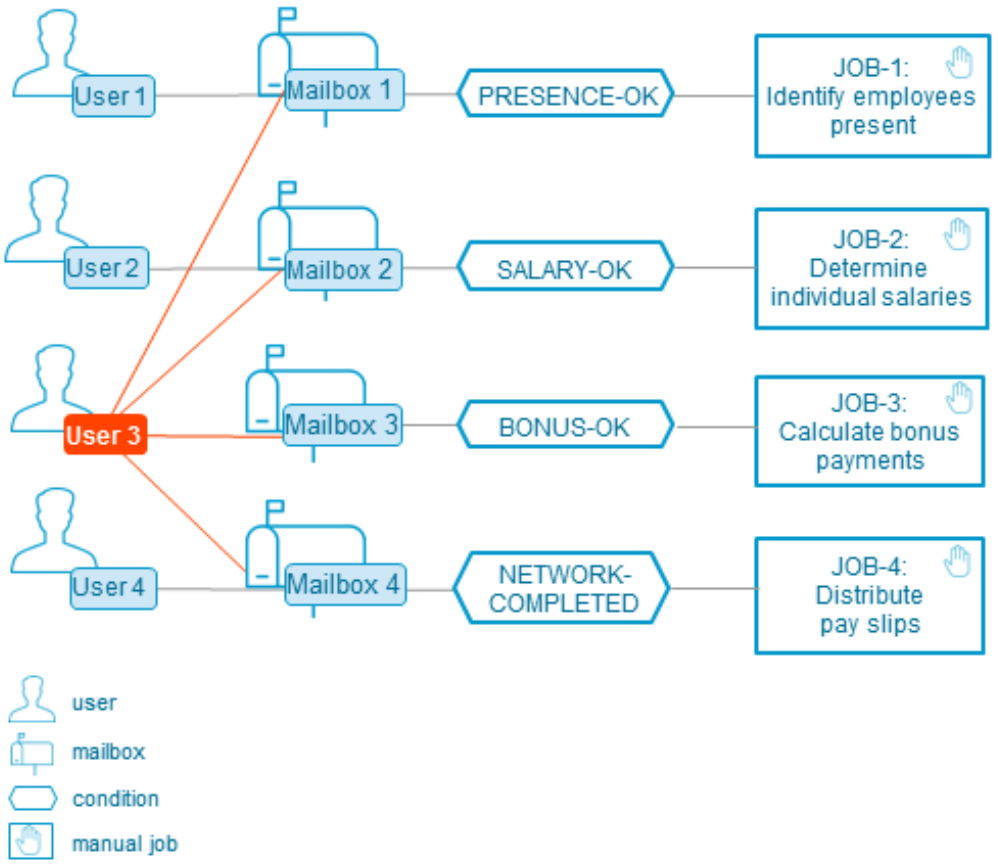
All users linked to a mailbox can display its list of outstanding messages and requests by issuing the direct command `MAIL`. With a single keystroke, the user is able to confirm the fulfillment of a condition or perform another requested action, like symbol prompting. After this the message vanishes from the list, the Entire Operations Monitor is automatically informed about the confirmation and triggers all jobs waiting for this condition.

See also the following sections:

- *Message Sending* in the *User's Guide*
- *Global Messages for Events* in the *Administration* documentation

Concept of Mailboxes

The following figure illustrates the concept of mailboxes using the example of the payroll department job network:



For example, User 1 (who could be in data collection) is notified that the condition PRESENCE-OK is not fulfilled. He can then take the necessary steps by completing personnel attendance data and confirming this in his mailbox, when finished. User 3 (who could be assistant to the personnel manager) is notified of any unfulfilled condition and can thus supervise the running of the whole job network. Users 3 and 4 are notified when the network has ended (NETWORK-COMPLETED) and the pay slips can be distributed.

Up to ten mailboxes can be associated with one user ID.

For more information, see the following documents:

- *Working with Mailboxes* in the *User's Guide*
- *Global Messages for Events* and *Mailbox Definition* in the *Administration* documentation

9 Resources

To make the availability of resources a prerequisite for job submission, Entire System Server functionality can be used.

Resources have to be defined with the System Administrator Services function (see the *Administration* documentation) before they can be specified as prerequisites for a job. The user can define and modify any number of resources to regulate the job flow. Resources prevent jobs from running in parallel if all other prerequisites (e.g. time windows or logical conditions) are fulfilled.

For a detailed description of the resource concept, see the section *Resources* in the *User's Guide*.

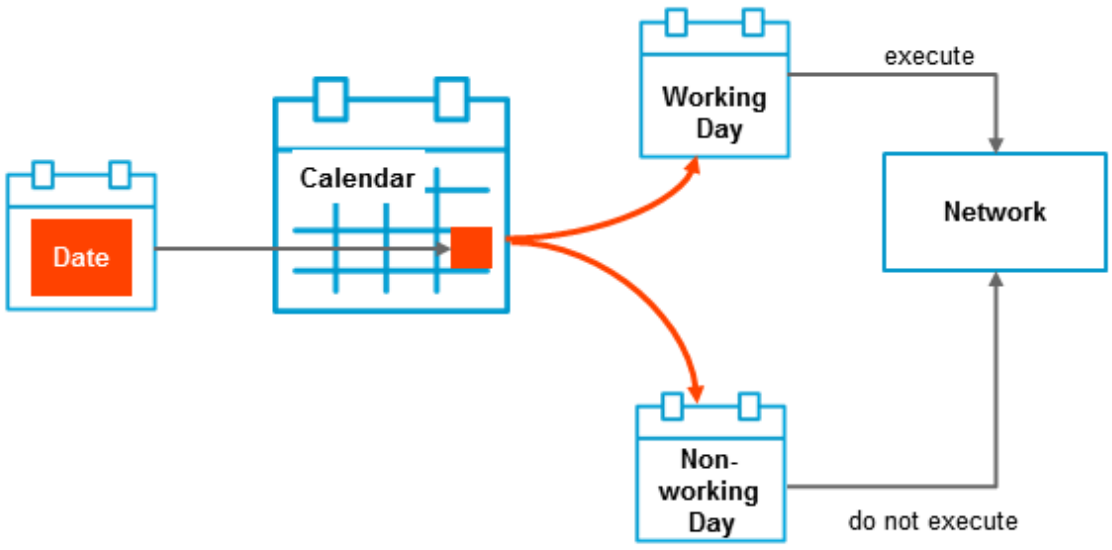
In our example of the payroll department job network (***Job Network with Logical Conditions***), we could prevent Job 6 (Generate Personnel Report) and Job 7 (Generate Pay Slips) from running in parallel by defining a resource (e.g. CPU time) with an initial quantity of 100 , and defining this resource for each job with a required value of 60. The jobs will then run sequentially.

10 Calendars

Calendars of any number can be defined to the system and easily modified online. They can apply to a dedicated owner or to the whole system.

The only purpose of having calendars in Entire Operations is to distinguish between working days and non-working days. Entire Operations does not perform any activities on non-working days. You can determine whether a job network scheduled for execution on a non-working day should be activated before or after the scheduled day or whether it should be cancelled.

On the other hand, if you want a job network to run, for example, each Friday, you do not need a calendar.



For more detailed information, see the sections *Calendars* and *Calendar Maintenance* in the *User's Guide*.

11 Schedules

Schedules contain the planned execution dates of job networks. They can contain periodic and/or explicit schedule dates as indicated in the following graphic:

You can define an unlimited number of schedules, and one schedule can be referenced in different job networks.

If a schedule table is based on a predefined calendar, execution dates can be defined relative to holidays (for example: the last working day of a month).

For more information, see the section *Schedule Maintenance* in the *User's Guide*.

12 Symbol Tables and Symbols

Symbol tables are user-defined tables, each containing a list of symbol names together with their current value. These tables are used during dynamic JCL generation. The benefit of using a symbol table is that it must be created only once, but can be referenced in a huge number of jobs.

You can define any number of symbol tables and use them just by specifying their name in the definition of the appropriate job networks.

Symbols can be defined for prompting during or before a job network activation. The data entered at the user's terminal is then assumed by the JCL to be executed.

Each network activation has its own active copy of the linked symbol table(s). This allows you to schedule networks with different parameter sets, even a long time in advance.

Any occurrence of a symbol name in the JCL or in a script is replaced by its current value from the symbol table. You can use two escape characters to determine whether this replacement should take place at JCL generation time or at job submission time.

There is also a large number of predefined symbols (see the *User's Guide*) available within Entire Operations.

Symbols are searched for in several tables (like in STEPLIBs) as described in *Symbol Table Types and Symbol Search Order* (*User's Guide*). Symbols can contain other symbols recursively; system variables can be used to construct symbol values.

A symbol table belongs to an owner. Any number of symbol tables can be associated with an owner. The user can update all symbol tables of all owners for whom he is authorized.

Symbols can be examined and modified by APIs (Application Programming Interfaces) from any Natural application. Scheduling such a program as part of an Entire Operations job network makes it possible to modify active symbol tables even during the execution of the job network.

For detailed information, see the section *Symbol Table and Symbol Maintenance* in the *User's Guide*.

13 Run Numbers

For more information, see the section *Run Number* in the *User's Guide*.

Run Numbers

Active Objects in Entire Operations are identified additionally by a run number, which is assigned to them automatically during the creation of an active object. Active networks or jobs are created during a network activation or job activation.

- Run numbers are unique on the job network level.
- Run numbers can be assigned to planned activations too. In the planning phase, no active jobs exist for a given run number.
- There is no guarantee that the numbering of network activation is ascending with the activation times.

Run Number Range

For a new network, the creation of run numbers starts with 1. If the highest allowed run number is reached, the numbering of active networks starts with 1 again. Imported Job Networks continue with their numbering from the previous environment.

The highest allowed run number can be defined with the System Administrator Services function (see the *Administration* documentation).

Reserved Run Numbers

Some run numbers are used by Entire Operations for special purposes. These numbers may not be used for any user networks and jobs.

Table of Reserved Run Numbers

Run Number	Equivalent	Usage
-1	`abs`	Internal representation of the condition reference `absolute`.
-2	`void`	Internal representation of the condition reference `void`.
-3	`K-RUN-MACROTEST`	Used during testing of Macro programs.
-4	`K-RUN-PREGENERATED`	Used for the storage of pregenerated JCL.

14 Object Versioning

With Entire Operations you can use different versions of objects of type **job network** and **symbol tables**.

This section describes the usage of this concept. For further information refer to the detailed description of the object types in the *User's Guide*.

Use of Versioning

The versioning of job networks and symbol tables is optional.

For each job network and for each symbol table you can decide individually whether you like to use versions, or not.

You may use versioning:

- to archive previous network versions, so that they can be activated manually any time later;
- to archive previous symbol table versions;
- to prepare new network versions or symbol table versions for future use.

By defining network version usage ranges or symbol table version usage ranges, you can define which version is to be used by scheduled activations.

Versioning of Job Networks

This section covers the following topics:

- Using Network Versions for Schedule Activation

Version Names

Version names can be a maximum of 10 bytes. The assignment of names is with few exceptions user-defined.

Following conventions, restrictions and recommendations apply:

- The use of upper and lower case characters is allowed.
- Space and the characters `?`, `<`, `>` are not allowed.
- Because of reserved names the usage of `,(` as first character of version names is prohibited.
- Due to the wildcard function the usage of "*" in a name is not allowed.
- To prevent problems during porting to other platforms do not use special characters and umlauts.

Version Names Exit

With the usage of a global version names exit you can force a customer-specific version name syntax. For detailed information, see *Global Exit for Version Names* in the *Administration* documentation.

Reserved Version Names for Networks

`<empty>`**; in selections and in the log also:** `(unnamed)`

Is used for an unnamed version.

This is the only existing network version, after migration from an Entire Operations version prior to Version 5.4.1.

In parameter listings "e.g. for Reporting" you can use also a hyphen "-".

`(current)`

Will be replaced by the version that is the set in the activated time schedule.

`(current)` can be used in version references.

Generate Network Versions via Cloning

You can use the copy function also to clone Job Networks and to create new versions.

This is a common way to build new network versions.

You can also use the import function to add a version.

Copying Jobs

Jobs in any version can copied out of the originating network.

Deleting Network Versions

- If multiple versions of a job network exist, then one must select one to be deleted.
- Unless the last, “or only” version is deleted, then automatically the “Network Main” object will be deleted too.
- Independent version objects that belong to the network will be deleted with the “Network Main” object.
- A network version can not be deleted if listed in a activated time schedule as a standard version. A defined date in the past is not relevant.

Deleting Network Versions or single Jobs via API

By using the API `NOPUAC5N` (Function `D`, Run Number `-1`) you can delete single network versions and jobs.

Using Network Versions for Schedule Activation

To maintain version usage in Network administration, use the corresponding functions described in *Maintaining the Usage of Network Versions* in the *User's Guide*.

This section covers the following topics:

- Maximum Number of Versions per Network

Evaluation and Activation of Network Versions

The following applies:

- If only one version exists in a network, then this version will be activated. A schedule definition will be ignored.
- If multiple versions exist in a network, then it will be checked if a version has a current activated time schedule. If this is so, then this version will be activated.
- If usage intervals are defined for a network version, but the activation date is not in this interval, then the network will *not* be activated although scheduled. Corresponding protocols "Log entries" and messages will be sent.

Manual Activation

If you choose a manual activation, then *any* network version can be selected. The standard version for schedule activation "if existent" will be offered to you first.

Activation of a Sub-Network

In the sub-network definition you can define any version or the reserved `(current)` name.

Activation as End-of-Job Action

For the network or job activation as end-of-job action you can define any version or the reserved `(current)` name.

Activation via API

For the network or job activation via API `NOPUAC5N` you can define any version or the reserved `(current)` name in the field `NETWORK-VERSION`.

Note that the API may issue version-related return codes.

Versions without Schedule Activation

In Entire Operations you can save multiple versions of job networks. Versions that are not, or are no more in the usage interval of schedule activation will not be activated automatically.

Daily History of Network Activations

The history of the network activations contains the network version for every run.

Reporting

Network versions will be considered.

Import / Export

Network versions will be considered.

Exit Functionality

Entire Operations exits that are active in networks, support network versioning.

Maximum Number of Versions per Network

The maximum number of network versions can be limited system wide as described in the section *Versioning of Job Networks*.

Versioning of Symbol Tables

This section covers the following topics:

- Version Names
- Version Names Exit
- Reserved Version Names for Symbol Tables
- Generate Symbol Table Versions via Cloning
- Copying single Symbols
- Deleting Symbol Table Versions
- Deleting Symbol Table Versions or single Symbols via API
- Using Symbol Table Versions for Schedule Activation
- Definition of Symbol Table Versions
- Active Symbol Tables
- Symbol Prompting
- Search Order for Symbols
- Symbol Tables at System and Owner Level
- Logging
- Cross References (XREF)
- Reporting
- Import / Export
- Exit Functionality

- Maximum Number of Versions per Symbol Table

Version Names

Version names can be a maximum of 10 bytes. The assignment of names is with few exceptions user-defined.

The following conventions, restrictions and recommendations apply:

- The use of upper and lower case characters is allowed.
- Space and the characters `?`, `<`, `>` are not allowed.
- Because of reserved names the usage of `,(,` as first character of version names is prohibited.
- Due to the wildcard function the usage of "*" in a name is not allowed.
- To prevent problems during porting to other platforms do not use special characters and umlauts.

Version Names Exit

With the usage of a global version names exit you can force a customer specific version name syntax. For detailed information, see *Global Exit for Version Names* in the *Administration* documentation.

Reserved Version Names for Symbol Tables

`<empty>`; in selections and in the log also: `(unnamed)`

Is used for an unnamed version.

This is the only existing network version, after migration from an Entire Operations version prior to Version 5.4.1.

In parameter listings "e.g. for Reporting" you can use also a hyphen "-".

`(current)`

Will be replaced by the version that is the set in the activated time schedule. (current) can be used in version references.

`(NV)`

Will be replaced with the network version of the active network. If only a unnamed network version exists, then this symbol table will be referred to this.

If a symbol table version with the same name is not existent in the network version, then a error message will be sent and the request aborted.

`(nv)` can be used in network versions.

`(svn)`

The symbol table version will be replaced thru the active network. `(svn)` can be used in referenced versions of a slaved network.

Usage for e.g.:

- job definition,
- all situations where `(svj)` can be defined.

`(svj)`

The symbol table version will be replaced thru the active job. `(svj)` can be used in referenced versions of a slaved job.

Usage for e.g.:

- Requested prerequisite dependant from the symbol value,
- Requested prerequisite dependant from multiple symbols,
- End-of-Job action: set symbol.

Generate Symbol Table Versions via Cloning

You can use the copy function also to clone symbol tables and to create new versions.

This is a common way to build new symbol table versions.

You can also use the import function to add a version.

Copying single Symbols

Symbols in any version can copied out of the originating symbol table.

Deleting Symbol Table Versions

- If multiple versions of a symbol table exist, then one must select one to be deleted.
- A symbol table version can not be deleted if listed in a activated time schedule as a standard version. A defined date in the past is not relevant.

Deleting Symbol Table Versions or single Symbols via API

By using the API `NOPUSY6N` you can delete single symbol table versions and symbols.

Using Symbol Table Versions for Schedule Activation

To maintain version usage in symbol table administration, use the functions described in *Maintaining the Usage of Network Versions* in the *User's Guide*.

Definition of Symbol Table Versions

The symbol table versions can be defined in the:

- network versions definition,
- job definition.

Active Symbol Tables

- The activation of symbol tables is a component of network and job activations.
- A symbol table can only be activated in a clearly identified version. The identification of the requested symbol table version is part of the activation process.
- Active symbol tables loose there version nomenclature `(current)` or `(nv)`. They are detached during activation.
- Active symbol tables can only have the version nomenclature `(none)` or a defined version name.
- If a requested symbol table version is missing, or the version cannot be defined, then the activation process will be aborted with a error message.
- If a situation is not clear, do not "guess" the symbol table version.

Symbol Prompting

Before symbol prompting (during manual activation and before executing the symbol prompting exit in the monitor), the symbol table versions to be used will be clearly determined. See also *Symbol Prompting during Network or Job Activation* in the *User's Guide*.

If at least one symbol table version cannot be identified, the activation process will be aborted with a error message.

Search Order for Symbols

The order in which symbols are searched for in the symbol tables defined in your environment depends on the hierarchy levels at which the symbol tables defined in you envionment can be accessed: see *Symbol Table Types and Symbol Search Order* in the *User's Guide* for details.

Symbol Tables at System and Owner Level

Symbol tables at system and owner level are not version-controlled.

The symbol tables are:

```
SYSDBA / A
<owner> / A
```

Logging

Logging of symbol actions include the version of the table where the symbol was loaded from.

The generated comments in the Entire Operations JCL header contains the symbol table version of all used symbols.

Cross References (XREF)

Symbol table versions will be considered.

Reporting

Symbol table versions will be considered.

Import / Export

Symbol table versions will be considered.

Exit Functionality

Entire Operations exits ans APIs that are related to symbols, support network versioning.

Examples:

- Symbol query-exit
- API `NOPUSY`*`x`*`N`

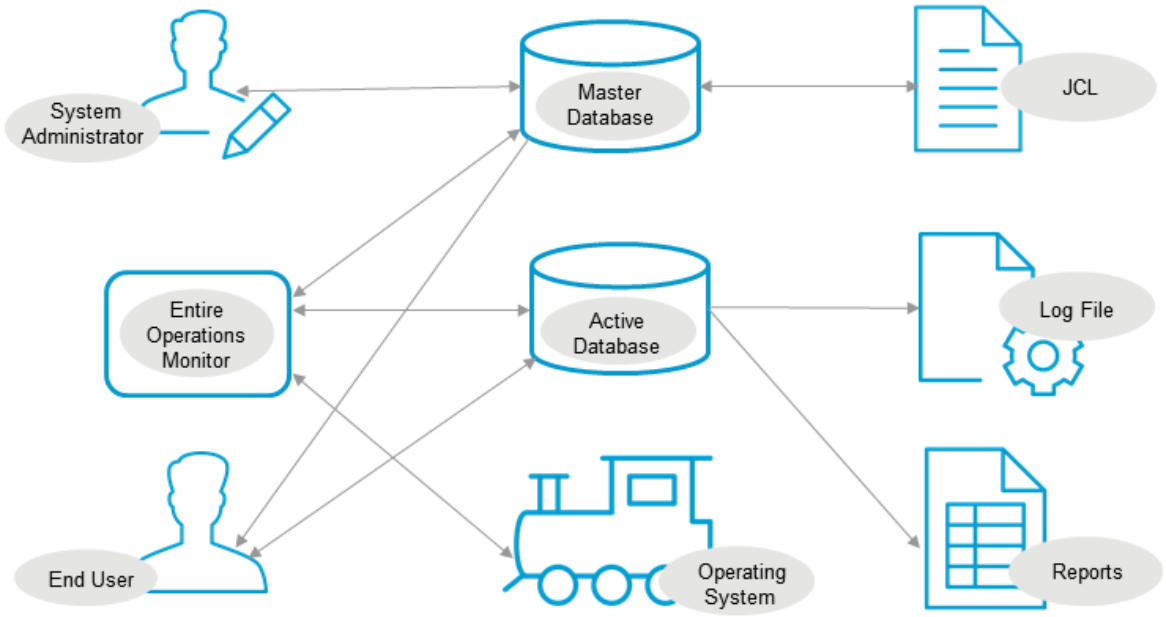
Maximum Number of Versions per Symbol Table

The maximum number of symbol table versions can be limited system wide as described in *Defaults for Network Options* in the *Administration* documentation.

III Entire Operations Components

15 Entire Operations Components

Entire Operations consists of the following components:



Master Database

All definitions and information concerning any user, job network, job, and scheduling information are stored on the master database. The master database is an Adabas file. This automatically provides features such as user synchronization, data integrity, data compression, auto-extension and auto-restart capability. The stored objects can be maintained online in any environment such as Com-plete, CICS, IMS TM, TSO, TIAM and *open*UTM as well as on UNIX. These objects are:

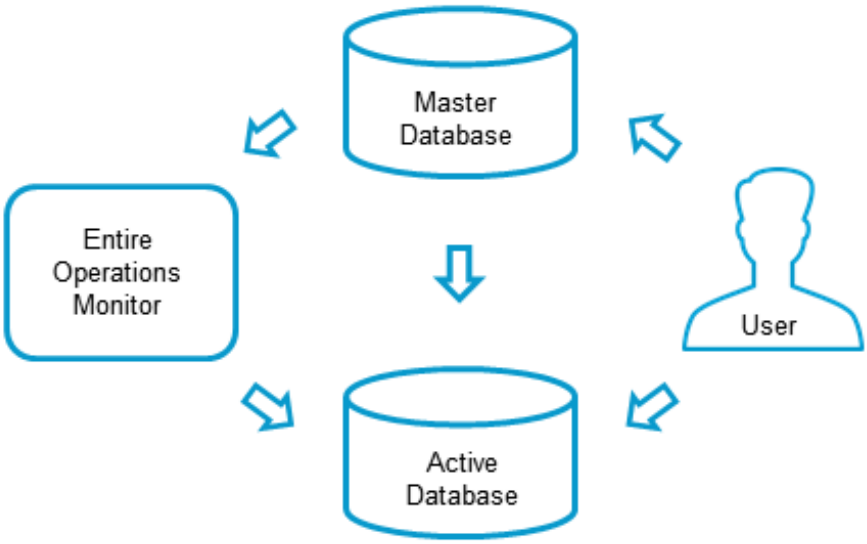
- User profiles;
- Job network definitions;
- Job definitions;
- Input and output conditions;
- Resource definitions;
- Schedule tables;
- Calendars;
- Symbol tables.

External JCL

Entire Operations can integrate JCL unchanged into job networks and the JCL can even remain in its original location. On z/OS, sequential data sets and PDS, LMS and z/VSE libraries are supported as well as the Natural and CA-LIBRARIAN storage types. On BS2000, the JCL can be located in SAM and ISAM files, or in LMS libraries. On UNIX operating systems, shell scripts can be included into job control by Entire Operations. On Windows, BAT files can be used.

The IMPORT function allows you to copy the JCL to the Master Database. This should always be used when required by particular backup criteria (access is then only possible through Natural Security) or if the JCL resides in the Master Database and is backed up with it.

Active Database



When a job network is activated (see the *User's Guide*), it is copied to the active database. A network is activated (see the *User's Guide*) either automatically by the Entire Operations Monitor according to its scheduled date, or manually by the user on demand. The active database may thus contain several copies of the same job network, each identified by a different run number.

The following information is stored:

- Current definition of scheduled job networks and their current symbol tables;
- Active JCL library (this means that all JCL information is copied from external storage media such as PDS, LMS, VSE-LIBRARIAN or UNIX files to the active database)
- Current status of input and output conditions;
- Current job status.

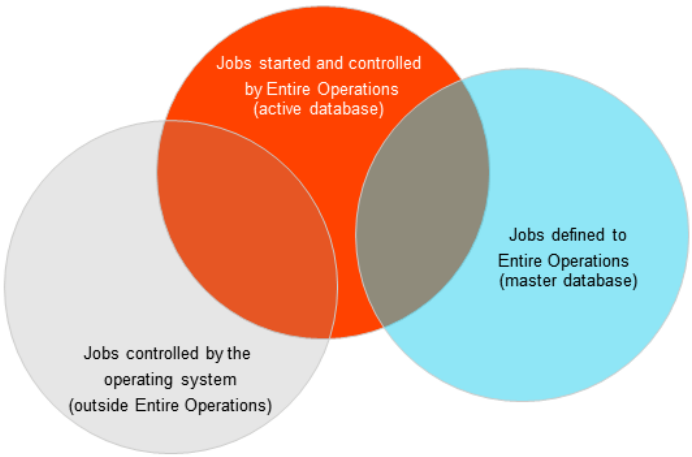
The active database can be accessed and its information modified in the same way as the master database. Changes to any object in the active database are only valid for the current run of the network and do not affect the network and job definitions stored on the master database. This enables you to make modifications which are valid only for a specific production cycle.

Basic Information on Active Networks and Jobs in the Active Database

At this point, all jobs are ready for execution according to their dependencies. They are said to be *in the active database*. However, not all jobs in the active database become operating system jobs: for example, dummy-type jobs or Natural programs are Entire Operations jobs which are not submitted to the operating system but executed by the Entire Operations.

You can distinguish between the following groups of jobs:

- **Jobs in the Entire Operations master database**
  (inactive Entire Operations jobs);
- **Jobs in the Entire Operations active database**
  (including jobs not submitted to the operating system);
- **Operating system jobs**
  (including jobs not defined to Entire Operations).



The orange circle contains all jobs started and controlled by Entire Operations. The blue circle contains all jobs defined to Entire Operations but not started. Entire Operations cannot control jobs that are not defined to it, and that are not started by it.

The intersection of the orange and blue circle indicates Entire Operation jobs that are ready to be started by Entire Operations.

Jobs outside Entire Operations (contained in the grey circle) run on the computer outside the control of the Entire Operations Monitor. On the other hand, Entire Operations holds information on jobs that run outside the control of the operating system.

The intersection of the grey and orange circle indicate Entire Operations jobs submitted to the operating system.

The active database is located on the Entire Operations active database and contains all operational information of the current run of the activated job networks.

You can access the active database to maintain active networks and jobs, including logical conditions, resources and scheduling parameters.

The following chapters describe how you can maintain active job networks and individual active jobs.

Entire Operations Monitor

The Monitor is the heart of Entire Operations. The Entire Operations Monitor is a complex program that is activated periodically and checks the definitions stored in the master database for any work to do. It activates and processes networks and jobs according to their prerequisites and controls running job networks, even if they reside on the nodes of different computers.

The Monitor performs the following functions:

- Automatically activates scheduled networks (copies them to the active database);
- Checks time windows for job or job network execution;
- Checks input conditions and resources;
- Submits jobs according to their (internal) priority;
- Keeps track of jobs in the various queues of the operating system;
- Analyses end-of-job status, determines which events have occurred and triggers appropriate system actions (set logical conditions, send messages, start programs);
- Logs all important information.

The functions of the Entire Operations Monitor can be distributed to various subfunctions (subtasks). Using subtasking, certain processing steps can run in parallel and multi-processor environments can thus be used to optimize performance. Distributing typical Monitor functions is the responsibility of the system administrator.

For further information, refer to the section *Monitor Defaults* in the *Administration* documentation.

Operating System

Entire Operations initiates job processing on the underlying operating systems: jobs, tasks, scripts or Natural programs are started. Several operating systems can be served simultaneously by one Entire Operations Monitor.

The Monitor sends the appropriate requests to the Entire System Server execution node. With Entire Net-Work, the Monitor can do this across the whole network.

Logging Facility

Entire Operations provides a logging facility which records every event during network processing. It also records all manual changes to active jobs, such as JCL corrections or updates of symbol tables. This information is available online and can be used for reporting and statistical purposes.

The log information can be selected according to job network and job, as well as according to dates and times, or even users. The log reports can also be exported into a sequential file, thus providing the opportunity for further analysis with any other tool and according to any criteria.

It is possible to define logging of JCL, job protocol and/or job-related messages for all or for some jobs.

For more information, refer to the section *Log Information* in the *User's Guide*.

Reporting Facility

A reporting facility provides a wide range of system information useful for keeping track of operations, for making scheduling decisions or even for planning future production cycles.

Reports are available on the basis of log information. They can be requested for all jobs or selected for a given date/time range, either for all terminated jobs or for abended jobs.

Descriptions of job networks are available in a brief outline and more detailed form.

A list of all jobs scheduled for a specific date can also be displayed. This makes it possible to forecast any future production date in advance.

All reports can be viewed online or printed for archiving purposes.

For more information, refer to the section *Reporting* in the *User's Guide*.

Editor

Entire Operations includes an adapted version of the Software AG Editor which can be used to create, display or modify any of the following:

- JCL in the master database, either in an external file or Natural member;
- JCL in the active database;
- Natural programs and user exits;
- Online documentation of networks, jobs and any events (e.g. scratch pad information);
- Job protocols and output (in browse mode).

Entire Operations Editor - Job Description

The Entire Operations editor allows you to handle all these data types in the same way in any operating system and run-time environment supported.

IV

16 Entire Operations Facilities

Entire Operations provides a number of facilities which allow the user to define objects to the system and control and monitor network processing.

17 Main Application Window

After starting the Entire Operations GUI Client, the Entire Systems Management application window opens providing all Entire Operations functions in the Object Workspace on the left and the content pane on the right.

For more information, see the section *Elements of the Main Application Window*.

18 Job Network Maintenance

Before Entire Operations can identify and activate a job network, this job network must be defined to the system. A short description can be added for easy identification when reviewing networks. A node number is specified to define the default machine on which all jobs in the network will run. A symbol table can be specified as well in order to use dynamic JCL creation feature. As the execution node, this name is used just as a default for all jobs contained in the job network and can be overwritten for each job.

For detailed information, see the section *Network Maintenance* in the *User's Guide*.

19 Job Network Scheduling

Scheduling a job network means defining dates and times at which it is to run. A job network is activated (copied into the active database) according to its scheduled starting time and additional schedule parameters defined for the network.

While processing a network, the Entire Operations Monitor checks whether all prerequisites for a job are fulfilled (time, resources, input conditions). According to the result it will automatically start the job and keep track of it.

For more information, refer to the sections *Defining a Network Schedule* and *Schedule Maintenance* in the *User's Guide*.

20 Job Maintenance

While in the job network maintenance facility, the user can request a list of jobs for a particular network. The user can add, delete and modify jobs in the network. A job is defined by (a logical) name, type and location (PDS member, Natural library etc.). A short description can be added for easy identification when reviewing the jobs in the network. Additionally, a node ID can be specified for execution to override the designated machine defined for the job network.

If the job is of the dynamically generated type, the symbol table used in dynamic generation must also be specified (see the section ***Dynamic JCL Generation***).

For detailed information, see the section *Job Maintenance* in the *User's Guide*.

21 Job Scheduling

The activation of each job depends on the scheduling parameters defined for it. As in the case of job networks, earliest starting time can be defined.

For jobs, however, the user can additionally specify whether a scheduled job that did not run (e.g. due to hardware problems) is to be rescheduled (i.e. the job is scheduled to run twice at the next scheduled time). A number of days can also be specified to determine how long a job can reside at most in the active queue before it is executed.

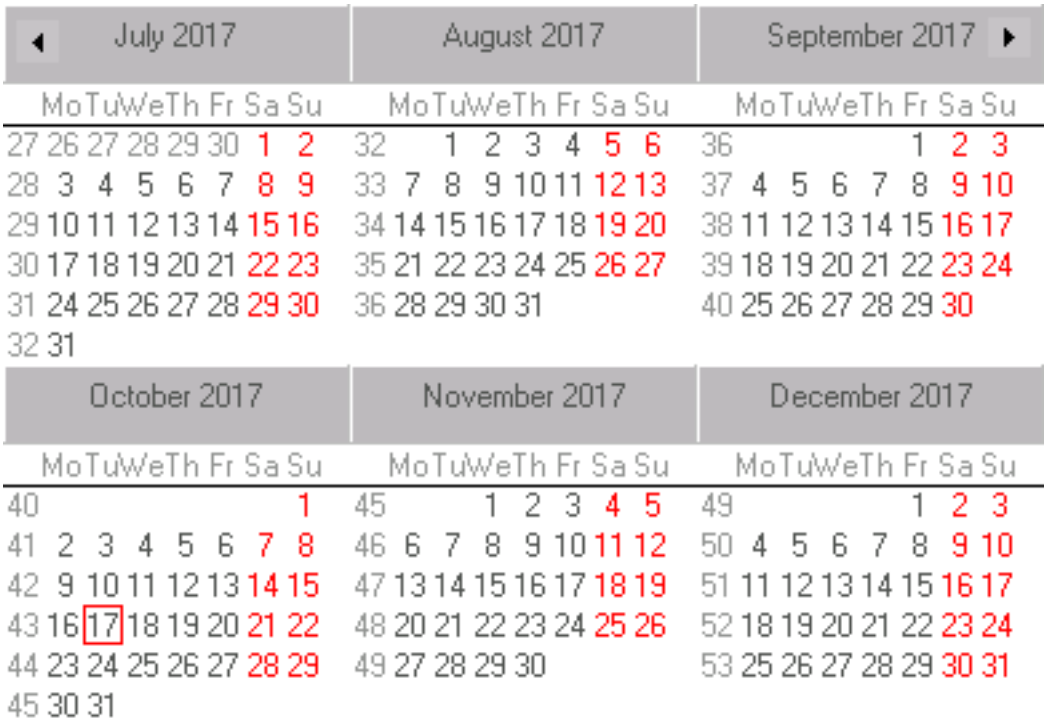
The user can also specify an estimated job running time which the Entire Operations Monitor uses to calculate starting and end time for the job. This can prevent a job from running if a predefined deadline would be exceeded.

The job scheduling parameters also provide a facility to notify specified users if the start of job fails to meet the defined deadline starting time.

For detailed information, see the section *Scheduling a Job* in the *User's Guide*.

22 Calendar Definition

Calendars are referenced by schedule tables which are defined in the network maintenance facility. Any number of calendars can be defined to the system. Calendars can belong to an owner or be used system-wide. In the calendar maintenance facility, the user can add, delete or update a calendar (system-wide calendars can only be modified by the system administrator).



July 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
27	26	27	28	29	30	1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30
32	31						

August 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
32		1	2	3	4	5	6
33	7	8	9	10	11	12	13
34	14	15	16	17	18	19	20
35	21	22	23	24	25	26	27
36	28	29	30	31			

September 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
36				1	2	3	
37	4	5	6	7	8	9	10
38	11	12	13	14	15	16	17
39	18	19	20	21	22	23	24
40	25	26	27	28	29	30	

October 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
40						1	
41	2	3	4	5	6	7	8
42	9	10	11	12	13	14	15
43	16	17	18	19	20	21	22
44	23	24	25	26	27	28	29
45	30	31					

November 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
45			1	2	3	4	5
46	6	7	8	9	10	11	12
47	13	14	15	16	17	18	19
48	20	21	22	23	24	25	26
49	27	28	29	30			

December 2017							
Mo	Tu	We	Th	Fr	Sa	Su	
49				1	2	3	
50	4	5	6	7	8	9	10
51	11	12	13	14	15	16	17
52	18	19	20	21	22	23	24
53	25	26	27	28	29	30	31

A calendar is defined by name and year. Defining new calendars or modifying existing ones consists of specifying or marking holidays (non-working days). The non-working days in the example above appear in in red.

Entire Operations accounts for holidays by not activating a network if the scheduled date is marked in the calendar as a holiday.

For detailed information, see the section *Calendar Maintenance* in the *User's Guide*.

23 Condition Maintenance

Logical conditions are dependencies between jobs. They are defined using the logical conditions maintenance facility. A logical condition can be added, deleted or modified. Any number of logical conditions can be assigned to any one job. A logical condition can have either of two statuses that determine how Entire Operations is to continue processing: TRUE (condition exists) or FALSE (condition does not exist).

All conditions are identified by name and a reference date to allow the Entire Operations Monitor to distinguish between the same event occurring on different dates. Dates can be specified as relative dates or explicit dates. All relative dates are converted to real dates when the job is put in the active queue. See also *Date and Time Formats* in the *User's Guide*.

There are two ways of using logical conditions:

- as **input conditions**;
- as **output conditions**.

Input Conditions

Input conditions must be fulfilled before Entire Operations can submit an active job. In order to link two jobs, an input condition must also be defined as an output condition for the preceding job. An input condition can be fulfilled by a CPU event or manually by the user.

Apart from a name and reference date, the user can also assign a mailbox to a condition. Each user ID can also be associated with up to 10 mailboxes. Entire Operations will automatically notify each user of all pending conditions assigned to any mailboxes associated with his user ID.

The user can also further specify what status the condition should be in before the job can be submitted (TRUE or FALSE), whether this job must wait until the condition applies exclusively to it (e.g. to prevent parallel running of two or more jobs with the same input condition), and whether Entire Operations is to automatically reset the condition after job submission.

Before job submission, all input conditions defined for the job are checked automatically by the Entire Operations Monitor. If you want the checking to be done by a Natural user exit this routine must also be specified in the input condition definition screen.

For detailed information, see the section *Defining and Managing Job Input Conditions* in the *User's Guide*.

Output Conditions

Output conditions will be maintained automatically by the Entire Operations Monitor if their associated events have occurred. In this case all jobs will be started which have these conditions as input conditions. Events and output conditions are defined within Entire Operations end-of-job checking (see the section ***End-of-Job Checking***).

As in the case of input conditions, output conditions are defined by name and reference. Additionally, the user can specify whether the output condition is to be set (to TRUE) or reset (set to FALSE) when the associated event occurs.

Up to 20 output conditions can be associated with a single event.

24 End-of-Job Checking and Actions

End-of-Job checking refers to the process of how Entire Operations recognizes the job status on job completion.

When the job has just completed Entire Operations searches for the occurrence of user-defined events. Such an event can be any of the following:

- A return code is received in a specific job step;
- A return code is received in any job step;
- A string is found in the job protocol or output;
- A Natural user exit is executed which determines the end-of-job status by returning a certain condition code. This routine can:
  - examine the job protocol or output itself,
  - read data produced by the job,
  - perform system functions,
  - send messages.

For detailed information, see the section *End-of-Job Checking and Actions* and *End-of-Job Checking and Actions: Columns EOJ Checking Page* in the *User's Guide*.

Default Checking

Depending on the operating system where the job was executed, Entire Operations performs some default checks to determine the job result. For z/OS systems, for example, system abends or JCL errors will automatically be detected. These default checks will be executed for each job, regardless, whether specific user-defined checks were requested for a job or not.

For detailed information, see the section *EOJ Checking Defaults for Various Operating Systems* in the *User's Guide*.

Retrying End-of-Job Checking

For the operating system z/OS the following applies:

- In case of incomplete SYSOUT, the SYSOUT reading will be retried 10 times, with intervals not shorter than 30 seconds. Interval can be longer if the monitor task wait time is longer.

End-of-Job Actions

For each specified event, the user can define how Entire Operations has to react. Such system action can consist of any of the following:

- Automatically set or reset the output conditions associated with this event (see the section ***Output Conditions***);
- Send a message (see the *User's Guide*) to a specific operating system user, the system console, an Entire Operations mailbox, Software AG's office automation system Con-nect, or to an e-mail address;
- Cancel or print job protocol and output;
- Perform a recovery (if a job or job step failed);
- Pass files to Entire Output Management (NOM) (see the *User's Guide*).

25 Using Resources

Resources can reflect real resources or they can be fictitious. The existence of real resources can be determined using Entire System Server features. For example you can examine the size of free space on any available disc, the presence of any cataloged data set or the actual number of running jobs.

Instead of this, resources are only meaningful within the Entire Operations system. An initial quantity of a resource can be defined in the system administration facility.

The user can use resources to further regulate the job flow.

For example, if a user defines a certain quantity of a resource as a prerequisite for a specific job, then this job will not run until this amount is available. The user could thus specify a combination of resources to define a certain job sequence within the job flow or prevent certain jobs from running in parallel.

For more information, see the section *Resources* in the *User's Guide*.

26 Dynamic JCL Generation

When defining a job within a network, a user can specify that its JCL is to be generated dynamically either at job activation time or at job submission time.

Dynamic JCL generation is achieved using the Entire Operations MACRO facility, an extension of the Natural programming language. This facility consists of standard Natural statements and text strings (JCL frames). The text strings can contain Natural escape characters followed by variables that will be replaced by their current value during dynamic generation.

These current values will be taken from **symbol tables** (see the relevant section) which must contain the current values to be substituted. The symbol table hierarchy to be used can be determined by choosing the **Usable Symbol Tables** function from the context menu of a network or job node. See also the sections *Listing Usable Symbol Tables* and *Symbol Escape Characters* in the *User' Guide*.

If any symbol specified in the dynamic JCL is not in the symbol table indicated for the job, the symbol is searched for at substitution time (either activation or execution) in the symbol table(s) belonging to owner SYSDBA. A user can define any number of entries in a single symbol table or any number of symbol tables.

Additionally, Entire Operations passes standard variables defined in the parameter section to the dynamically generated program, such as job owner, network name, current job name and original scheduling date. The same applies to Natural system variables such as *DATE, *TIME and *USER. As these parameters can be replaced in any part of the JCL, different JCL configurations can be generated depending on time, date, user ID etc.

For further information, see the sections *Dynamic JCL Generation (JCL Location MAC)* and *Editing Macro Sources for Dynamic JCL Generation* in the *User's Guide*.

Entire Operations provides dynamic JCL generation for all supported operating systems (z/OS, z/VSE, BS2000, UNIX and Windows) as shown in the following examples.

Example 1: Dynamic JCL in a z/OS Environment

The following is the symbol table specified for the MACRO program:

Symbol Name	Current Value
STEPLIB	SN.SYSF.SOURCE
CLASS	G

The variable from the parameter section is assumed to have the following value:

P-OWNER	NET1

The system variables are assumed to have the following values:

*TPSYS	COMPLETE
*DEVICE	BATCH
*INIT-USER	SN

The following is a Natural MACRO program including a parameter section and JCL with the Natural escape character (#) followed by variable names from the symbol table:

```
# DEFINE DATA PARAMETER USING NOPXPL-A
# LOCAL /* MUST BE CODED
# END-DEFINE
//SNMAC4 JOB ,#P-OWNER,MSGCLASS=X,CLASS=#CLASS //STEP01 EXEC
PGM=NOPCONTI,PARM='C0004' //STEPLIB  DD DISP=SHR,DSN=#STEPLIB
//* DEVICE: *DEVICE, INIT-USER: *INIT-USER //* TPSYS: *TPSYS
# IF CLASS = 'G'
//* THE MSGCLASS IS REALLY 'G'
# ELSE
//* ANOTHER MSG-CLASS FOUND
# END-IF
//*
```

The resulting dynamically generated JCL will be:

```
//SNMAC4 JOB ,NET1,MSGCLASS=X,CLASS=G
//STEP01 EXEC PGM=NOPCONTI,PARM='C0004' //STEPLIB  DD
DISP=SHR,DSN=SN.SYSF.SOURCE //* DEVICE: BATCH, INIT-USER: SN
//* TPSYS: COMPLETE
//* THE MSGCLASS IS REALLY 'G'
//*
```

Example 2: Dynamic JCL in a BS2000 Environment

The fields taken from the DB-INFO are assumed to have the following values after the FIND statement:

Field	Value
NUCLEUS	055
LP1	1000
NU1	100
ACCOUNT	EXAMPLE
NH1	4000
MSG	FHL
VERSION	524

The variables taken from the parameter section have the following current values:

Variable	Value
P-OWNER	OS
P-JOB	NUC055
P-EXECUTION-NODE	055

No symbol table was defined for this example job.

The following is the example JCL written using the Natural MACRO facility, including variables to be substituted from the DB-INFO view and the parameter section. Variables are preceded by the escape character (#):

```
# DEFINE DATA PARAMETER USING NOPXPL-A
#  1 L-JOB
# 1 REDEFINE L-JOB
# 2  L-JOB-A    (A3)
# 2  L-JOB-NUC  (N3)
# LOCAL   /* LOCAL VARIABLES START HERE
# 1 DB-INFO VIEW OF DB-INFO
# 2  NUCLEUS
# 2  LP1
# 2  NU1
# 2  ACCOUNT
# 2  NH1
# 2  MSG
# 2  VERSION    /* E.G. 524
# 1 LWP  (N7)
# 1 NUC  (N3)
# 1 SPOOL  (A10) INIT <'NOSPOOL'>
# END-DEFINE
# *
# MOVE P-JOB TO L-JOB-A
# MOVE P-EXECUTION-NODE TO NUC
# F1. FIND DB-INFO WITH NUCLEUS = NUC
/.NUC NUC LOGON #P-OWNER,#ACCOUNT
/OPTION MSG=#MSG
```

```
/REMARK
/REMARK  NUCLEUS #NUC
/REMARK
/SYSFILE  SYSLST = NUC NUC..LST.NUC
/SYSFILE  SYSDTA = SYSCMD
/FILE  ADA VERSION..MOD,LINK=DDLIB
/FILE  *DUMMY,LINK=DDLOG
/FILE  *DUMMY,LINK=DDSIBA
/FILE  ADA NUC..ASSO,LINK=DDASSOR1,SHARUPD=YES
/FILE  ADA NUC..DATA,LINK=DDDATAR1,SHARUPD=YES
/FILE  ADA NUC..WORK,LINK=DDWORKR1,SHARUPD=YES
/EXEC  (ADARUN,ADA VERSION..MOD)
# COMPUTE LWP = F1.LP1 * (F1.NU1 + 100)
ADARUN PROG=ADANUC,LP=F1.LP1,LU=65535,LWP=#LWP ADARUN
DB=#NUC,NU=#NU1,NC=20,TT=600,TNAE=1800 ADARUN NH= NH1
/SYSFILE  SYSLST = (PRIMARY)
/SYSFILE  SYSDTA = (PRIMARY)
/SYSFILE  SYSOUT = (PRIMARY)
/LOGOFF  SPOOL
# END-FIND
```

The resulting dynamically generated JCL will be:

```
/.NUC055 LOGON OS,EXAMPLE
/OPTION MSG=FHL
/REMARK
/REMARK  NUCLEUS 055
/REMARK
/SYSFILE  SYSLST = NUC055.LST.NUC
/SYSFILE  SYSDTA = SYSCMD
/FILE ADA524.MOD,LINK=DDLIB
/FILE *DUMMY,LINK=DDLOG
/FILE *DUMMY,LINK=DDSIBA
/FILE ADA055.ASSO,LINK=DDASSOR1,SHARUPD=YES
/FILE ADA055.DATA,LINK=DDDATAR1,SHARUPD=YES
/FILE ADA055.WORK,LINK=DDWORKR1,SHARUPD=YES
/EXEC (ADARUN,ADA524.MOD)
ADARUN PROG=ADANUC,LP=1000,LU=65535,LWP=200000 ADARUN
DB=055,NU=100,NC=20,TT=600,TNAE=1800 ADARUN NH=4000
/SYSFILE  SYSLST = (PRIMARY)
/SYSFILE  SYSDTA = (PRIMARY)
/SYSFILE  SYSOUT = (PRIMARY)
/LOGOFF NOSPOOL
```

Note: Any JCL generated at activation time using the MACRO language can be modified by the user until the job is actually submitted. Of course this modification is valid only for the current network run.

Example 3: Dynamic JCL in a UNIX Environment

The following example illustrates dynamic symbol replacement within a Bourne shell script (escape character §):

```
#
# Bourne shell script for checking the number of users
# entered in /etc/passwd.
# If more than §USER-LIMIT entries appear,
# the script will be ended with exit 1.
#
#!/bin/sh
set -x
USER_COUNT='wc -l < /etc/passwd'
echo Number of users on node 'hostname' : $USER_COUNT
if test $USER_COUNT -gt §USER-LIMIT
then
    echo USER_COUNT_WARN
    exit 1
else
    echo USER_COUNT_OK
fi
```

The symbol table to be used should appear as follows:

Symbol Name	Current Value
USER-LIMIT	100

The result is the following executable shell script:

```
#
# Bourne shell script for checking the number of users
# entered in /etc/passwd.
# If more than 100 entries appear,
# the script will be ended with exit 1.
#
#!/bin/sh
set -x
USER_COUNT='wc -l < /etc/passwd'
echo Number of users on node 'hostname' : $USER_COUNT
if test $USER_COUNT -gt 100
then
    echo USER_COUNT_WARN
    exit 1
else
    echo USER_COUNT_OK
fi
```

Note: Any JCL generated at activation time using the Natural MACRO language can be modified by the user until the job is actually submitted. Of course this modification is valid only for the current network run.

27 Editing System Objects

The editing facility provided by Entire Operations allows the user to create, browse or edit any of the following objects:

- JCL of jobs, either in the master database (and thus from any external storage source) or in the active database. Changes to the JCL of active jobs are valid for the current run only and do not affect the master database;
- Natural programs and user exits (see *Editing JCL or Natural Programs* for more information);
- Entire Operations MAC (Macro) jobs (see *Editing Macro Sources for Dynamic JCL Generation* for more information);
- Online documentation of a network, job or an event within a job (scratch pad information);
- Job protocol (for browsing only);
- Job output (for browsing only).

This enables the Entire Operations user to process such various data as, for example, UNIX scripts, CA-LIBRARIAN files or LMS files with a single editor.

The user can enter the editor by selecting the editing option in the maintenance screens of the appropriate object.

The editor provides a comprehensive range of z/OS ISPF-like commands appropriate to the particular object to be edited. For example, the STOW command will save and catalog a Natural program, but should not be used when editing a Natural MACRO program. A Natural MACRO program is saved, pre-processed and cataloged by the MACRO command.

Other word processing facilities provided by the editor include centering, physical and logical tabulation, and text overlay.

28 Generating Reports

Entire Operations provides a wide range of reports to assist operations at all levels. The user can enter the reporting facility either by selecting the reporting option or by issuing the REPORT direct command. The reports cover the following:

- Information on all jobs, selectable according to terminated jobs, abended jobs and jobs which have not been started. Date ranges and network names can be specified to further narrow selection down. Reports on all jobs include all events, activation time, messages, termination status etc. All job reports are sorted according to log time.
- Network information, selectable according to whether a short overview is required or a longer report that includes detailed information on network components. In all network reports, information is provided on network and job definition, all input conditions and resources, as well as end-of-job handling, including output conditions. The more detailed reporting option additionally displays all text descriptions available on the network, job and event level.
- Schedule overview of selected or all networks, consisting of a list of jobs to be scheduled within a specified range of dates. This can either be done for time ranges belonging to the past to get a list of all unsatisfied network activations or you can request the Network schedule overview for future production periods in order to get information for forecasting and planning purposes.

All reports are available online, their data can be printed as well. This provides an easy form of long-term documentation.

Using the Entire Operations Import/Export Utility, the contents of the master database can be unloaded to a sequential file, too. This feature is intended for migration and transport purposes, but you can also use it to build up your own reporting system.

For detailed information, see *Reporting* in the *User's Guide*.

29 Cross References

Entire Operations provides functions to obtain cross reference information on the use of Entire Operations objects.

For detailed information, see *Cross References* in the *User's Guide*.

30 Message Sending

Entire Operations can send messages to various locations. This can be triggered by system-detected events and user-defined events.

For more detailed information, see the section *Message Sending* in the *User's Guide*.