

English



FUJITSU Software BS2000

AVAS V8.5A

AVAS Functions and Tables

User Guide

Edition June 2017

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH

www.cognitas.de

Copyright and Trademarks

Copyright © 2017 Fujitsu Technology Solutions GmbH.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Contents

1	Preface	7
1.1	Objectives and target groups of this manual	8
1.2	Structure of the AVAS documentation	9
1.3	Changes since the last edition of the manual	10
1.4	Notational conventions	11
1.5	Licensing regulations	12
2	Overview of AVAS functions	17
3	Production definition	25
3.1	Nets for production run	26
3.1.1	Net description	26
3.1.2	Creating and editing nets	31
3.1.3	Defining BS2000 jobs, S procedures, FT requests and subnets	35
3.1.4	Time specifications in nets	36
3.1.5	Defining conditions in net descriptions	40
3.1.6	Defining and editing hypernets and subnets	46
3.2	Jobs for the production run	48
3.2.1	Creating and editing jobs	48
3.2.2	AVAS statements and variables in jobs and JCL elements	50
3.2.3	Creating the AVAS user masks	52
3.2.4	Editing jobs and JCL elements	53
3.2.5	Editing documentation elements	54
3.3	Restarts in nets	55
3.3.1	Preparing for the restart	55
3.3.2	Restarts in hypernets and subnets	66

Contents

3.4	Time scheduling	68
3.4.1	Calendars	68
3.4.2	Periods	70
4	Production execution	71
4.1	Production planning	71
4.1.1	Planning standard nets	71
4.1.2	Linking symdats	73
4.1.3	Planning hypernets and subnets	75
4.2	Production preparation	77
4.2.1	Modifying standard nets	78
4.2.2	Time of modification	83
4.2.3	Modifying hypernets and subnets	84
4.3	Release for production	85
4.3.1	Release of nets for processing	86
4.3.2	Connecting AVAS and MAREN	87
4.4	Production control and execution	88
4.4.1	Run control for standard nets	89
4.4.2	Run control for hypernets	92
5	Production monitoring	97
5.1	Monitor	98
5.2	Journal file	98
5.3	SYSOUT file	101
5.4	History file	102
5.5	Runtime logs	103
5.6	Reports	105

6	Status of the networks during processing	107
6.1	Overview of net statuses for the statements	107
6.2	Overview of log statuses for the statements	116
6.3	Overview of the status of the condition descriptions for each statement	117
7	Multiprocessor operation with AVAS	119
7.1	BS2000 multiprocessor operation with AVAS	119
7.2	Controlling remote systems with AVAS-SV-BS2	124
7.2.1	Controlling and monitoring of server jobs	124
7.2.2	Monitoring of the remote systems/servers	127
7.2.3	Tools for displaying and managing decentralized systems/servers	129
8	Automation of net runs	131
8.1	BATCH statements	131
8.2	AVAS program interface	132
9	Administration	133

- 10 Tables and overviews 139**

- 10.1 AVAS statements and parameters 139**
- 10.2 AVAS statements via /INFORM-PROGRAM 145**
 - 10.2.1 Operating the run control system 145
 - 10.2.2 Controlling released nets via calling the run control system 145
 - 10.2.3 Access to the SYSOUT/SYSLST of the AVAS processes 146
- 10.3 AVAS operations and operation characters 147**
- 10.4 AVAS statements and mask sequences 150**
 - 10.4.1 Overview of AVAS statements and mask sequences 150
 - 10.4.2 Representation of the mask sequences with NET-CONTROL 180
- 10.5 Restart prompting 197**
- 10.6 Net statuses for the statements 198**
- 10.7 Status of the condition descriptions for statements 206**
- 10.8 Log statuses for statements 208**

- Glossary 209**

- Related publications 221**

- Index 223**

1 Preface

The complexity and workload of computer centers are constantly increasing. DP operations therefore require clear structuring, high levels of transparency and flexibility, and constant productivity enhancement. The automation of batch production is an essential factor to reach this aim.

The AVAS (in German: **Auftragsverwaltungs- und Abwicklungssystem**) job administration and processing system provides computer center operators with a means of automating their job production to such an extent that operator intervention is reduced to an absolute minimum. Transferring batch processing into system layers that do not require operator intervention is greatly facilitated.

AVAS automates planning, preparation, release, control and monitoring of batch job processing in BS2000. The AVAS administration and control functions also run in BS2000.

From the BS2000 platform AVAS can start and monitor jobs on other systems:

- In the homogeneous BS2000 multiprocessor network AVAS uses the HIPLEX MSCF functions for job distribution and monitoring.
- The use of the AVAS-SV-BS2 server enables Remote BS2000 systems to be connected to AVAS via the Socket interface

The transfer of files to other vendors' systems is supported with an openFT connection.

In all cases the definition, preparation and monitoring of production is performed centrally by AVAS on a BS2000 system.

AVAS enables the computer center to automate its production jobs and to handle the necessary planning, preparation and monitoring tasks interactively. AVAS supports both decentralized work scheduling in the various non-DP departments and the central storage of information on jobs.

The *net description* defines the arrangement of jobs in the net, timing specifications, job characteristics, restart variants, and dependencies on other nets and jobs, and on condition values and resources.

Time scheduling uses calendars with symbolic dates or procedure names which, together with the net descriptions, form a *production plan*.

During *production preparation*, it is possible to supply the nets with runtime parameters from the production plan via user masks or from parameter files.

During the *release for production*, transport lists and tape mount listings can be created for the required data volumes by accessing the MAREN catalog. After the net has been released, it is started by the run control system in the *production execution* stage in accordance with the predefined time specifications and dependencies.

Production monitoring, like all of the preceding steps, is carried out online. If an error occurs, predefined restart processing is activated. Depending on the specifications in the net, restart is either executed automatically or initiated explicitly by the user, the latter permitting further manual intervention. The entire production sequence within the planned nets is logged and can be reconstructed on the basis of the journal. Under AVAS, the runtime logs of jobs can be saved and displayed.

1.1 Objectives and target groups of this manual

This manual addresses AVAS users and the AVAS administrator.

1.2 Structure of the AVAS documentation

The following documentation is available for the AVAS software product under the BS2000 operating system:

AVAS Functions and Tables

The “**AVAS Functions and Tables**” manual is intended for AVAS users. It initially provides an overview of the AVAS functions, and then a detailed description of how to define and handle production runs. The manual also includes brief descriptions of multihost operation and of administration, plus tables and overviews.

AVAS Statements

The “**AVAS Statements**” [1] manual is intended for AVAS users and the AVAS administrator. It contains all the AVAS statements in alphabetical order. The masks are described together with the corresponding AVAS statements.

The manual also contains information on

- conducting a dialog
- preparing jobs for execution under AVAS
- the CHECK function.

AVAS for the Administrator

The system administrator guide “**AVAS for the Administrator**” [2] is intended for those responsible for administrating the AVAS system. This manual describes all the tasks performed by the AVAS administrator, from generating the system to the administration of the AVAS system. The manual also includes information on

- the utility routine AVAS-QUER
- combining AVAS with MAREN
- AVAS reports
- BATCH functions
- external creation of AVAS elements
- program interface
- AVAS-SV-BS2

You can find these manuals online at <http://manuals.ts.fujitsu.com> or you can order them in printed form for a fee at <http://manualshop.ts.fujitsu.com>.

Readme file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at <http://manuals.ts.fujitsu.com>. You will also find the Readme files on the Softbook DVD.

Information under BS2000

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

```
SYSRME.<product>.<version>.<lang>
```

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at <http://manuals.ts.fujitsu.com>.

1.3 Changes since the last edition of the manual

The following change has been made since the last edition of the manual:

The connection of other vendors' systems (Linux, Windows etc.) with AVAS-SV is no longer supported.

1.4 Notational conventions

References to other publications are specified in the text using abbreviated titles. The full title of each publication, which is referenced by a number, can be found under “Related publications” after the corresponding number.

Cross-references within this manual indicate the number of the relevant page in the manual and, if appropriate, the name of the section or chapter. References to information described in other manuals include only the abbreviated title of the corresponding manual. You can use the index of the manual indicated to find the appropriate place in the text.

Supplementary information appears under the heading “Notes”.

The following notational conventions are used in this manual:



This symbol denotes important information which you should always observe.



This symbol and the word **CAUTION!** precede warning information. In the interests of system and operating security you should always observe this information to prevent the loss of data.

`fixed`

Path and file names as well as phase names and procedure examples are displayed using a `fixed` font.

bold
`fixed`

Entered by the user on the screen.

1.5 Licensing regulations

The licensing regulations for the OpenSSL package and the TLS-FTP patch of Peter 'Luna' Runestig are printed below.

LICENSE ISSUES

=====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts.

OpenSSL License

```
/* =====
 * Copyright (c) 1998-2016 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
```

```
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/
```

Original SSLeay License

```
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
```

```
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*   notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in the
*   documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
*   must display the following acknowledgement:
*   "This product includes cryptographic software written by
*   Eric Young (eay@cryptsoft.com)"
*   The word 'cryptographic' can be left out if the routines from the library
*   being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*   the apps directory (application code) you must include an
*   acknowledgement:
*   "This product includes software written by Tim Hudson
*   (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

```
/*
 * Copyright (c) 1999 - 2002 Peter 'Luna' Runestig <peter@runestig.com>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modifi-
 * cation, are permitted provided that the following conditions are met:
 *
 * o Redistributions of source code must retain the above copyright notice,
 *   this list of conditions and the following disclaimer.
 *
 * o Redistributions in binary form must reproduce the above copyright no-
 *   tice, this list of conditions and the following disclaimer in the do-
 *   cumentation and/or other materials provided with the distribution.
 *
 * o The names of the contributors may not be used to endorse or promote
 *   products derived from this software without specific prior written
 *   permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LI-
 * ABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUEN-
 * TIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEV-
 * ER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABI-
 * LITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
 * THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

2 Overview of AVAS functions

The AVAS functions are subdivided into the following two areas:

- production definition (job management) i.e.
 - generation of the requisite data structures such as net and job descriptions,
 - documentation,
 - parameters and the time schedule for processing under AVAS.
- production execution (job handling), i.e.
 - planning of the production processes,
 - production of the nets and the associated jobs,
 - release of the nets,
 - implementation of the production run at the scheduled time on the basis of the data structures generated.

Creating nets and jobs

Logically related jobs are combined to form job nets for execution under AVAS. A job net consists of 1 to n jobs, which may be defined for BS2000 platforms or formulate FT requests. BS2000 jobs can be either typical ENTER jobs or SDF-P S procedures.

Instead of complex controls that define the sequence of jobs, AVAS forms transparent nets that take account of all logical and temporal dependencies.

The user can insert parameter placeholders for variable job data within each job.

On all platforms, the user can use a statement in the JCL(Job Control Language) of the jobs to request AVAS to transfer the job log to AVAS management once the job has been executed. As FT requests include no JCL, no job log is taken into account here.

The processing sequence and the requirements for starting individual steps must be defined in the net structure. Dependencies on other nets, jobs, conditional values, or resources can be specified in the net structure in the same way.

Separate documentation elements can be defined for the documentation of nets, jobs and dependencies.

Provisions for the restart of the net after an error are incorporated in the net structure and in the jobs.

Nets with jobs and conditions are referred to as standard nets.

Groups of nets and jobs are combined to form what is known as a hypernet.

Real or symbolic start dates can be entered in the nets and job descriptions for the purposes of time scheduling.

Time scheduling

The time schedule comprises a calendar with a day column containing real dates and symbolic start dates that are assigned to the individual calendar days. Certain calendar days can be declared as non-working days using a special date.

The calendar and the job nets are linked by means of symbolic start dates. Which subset of jobs and dependencies are to be scheduled for this date is also defined using a symbolic start date. Multiple symbolic start dates can also be linked, for example "WT - MON", if a net is to be planned for every work day except Monday.

This provides the foundation for the long-term planning of DP production.

Periods, which specify a planning period within the calendar, can be defined to facilitate the planning of production runs for shorter periods of time.

Production planning

A production plan must be set up for the computer center for the actual production run for a defined period of time. The planning period is freely selectable. The AVAS production plan can be used to determine which nets are run on which days, and their current stage of preparation.

During production planning, the nets are linked with a real start date, supplied with variable data for the actual production run, and incorporated in the production plan. Subnets are planned in the context of a hypernet. Planning of nets and subnets is carried out either using the calendar, or in the case of an individual net, without the calendar.

The defined periods can also be used as the planning periods (planning via the calendar).

Networks can also be planned cyclically for a specified period.

Production preparation

An important task performed during production preparation is updating values that vary from job to job. A distinction between jobs that are modified regularly and jobs that can be executed without modification is therefore made in the net structure descriptions. FT requests are not modified.

The parameters can be updated by either entering the variables for a job or a net using a mask that has been assigned to the job or net, or by means of a parameter file. AVAS also provides predefined system variables such as, for example, Date and Time. The AVAS administrator can also define installation-specific AVAS system variables.

For fully-automatic production preparation, the names of the files that contain the up-to-date parameters can be made a fixed part of the net description. This allows users to supply the parameter files with the modified values and have AVAS perform production automatically through upstream computerized processes.

During production preparation, CC-defined routines can be used to define values in the input masks and to check any values entered for plausibility. This contributes to the automation process and considerably improves the quality of the production preparation process.

Subnets are produced within the context of the hypernet.

Release for production

The nets must be released by production preparation before net processing can be started. Processing can be performed according to the production plan using predefined time periods.

It is also possible to release single nets for processing. If the interfacing module to MAREN is being used, all the volumes required by the net to be released are checked to see whether they are available. If necessary, the VSNs are entered in the JCL. Volume readying lists are generated.

Subnets are released within the context of the hypernet.

Production execution

Production execution is initiated and controlled under AVAS by the run control system of the AVAS system. After release, the individual jobs are scheduled automatically.

The run control system starts the nets at the specified times, queuing the jobs in the order defined in the structure description and taking into account all declared dependencies on other nets, jobs, job variables, conditional values, and resources.

If an error occurs, the restart variants defined in the net description are initiated either automatically by AVAS or at the request of the user. The run control system processes the structure planned for the purpose of a restart automatically.

If a job or the AVAS net terminates abnormally, AVAS can branch to a data center routine in which the AVAS system administrator can provide error recovery sequences, particularly for low operator intervention and unattended data center environments. In the event of the abnormal termination of strategic jobs, it is thus possible to call a standby service or, in the case of dependent procedures, a replacement strategy.

Production monitoring

Net processing can be monitored at all stages of the production run. Thus, the progress of an individual net or of the entire production run can be monitored. Similarly, monitoring can also encompass all nets with a certain processing status.

The journal file allows all the DP production runs handled by the AVAS system to be monitored for control purposes. This file contains all user activities and all actions of the AVAS system for the processed nets. A history file provides compressed historical data such as, for example, average runtimes of the nets and jobs. It is also possible to show the job runtime logs which have been taken over from AVAS.

AVAS reports

In the same way as with the AVAS dialog system, it is possible to load a process – the REPORT generator. The REPORT generator allows you to create AVAS reports and output them to a print file, the report file.

An AVAS report represents an evaluation of the AVAS production plan and the AVAS journal file according to specified criteria. The user defines these criteria with REPORT statements which the REPORT generator reads from the BS2000 system file SYSDTA.

The PLANNED-NET-MODIFICATION report lists nets which have been changed subsequent to production planning.

The OUT-OF-PLAN lists nets which have been delayed by more than a defined amount and/or are in a selected status.

The quantity of nets to be logged is determined on the basis of the current production (NPRLIB). It can be defined more precisely by specifying NET-NAME and/or PERIOD-NAME. The data to be logged is selected from the current journal file (JRNDAT).

AVAS-QUER

The AVAS-QUER utility routine reads the AVAS data stock in BS2000 and selects data for further processing in relational databases.

In AVAS V8.0 and higher the scope of the selected data can be controlled using the TABLE-STRUCTURE parameter. This enables all the data which is required for automatic creation of an AVAS net to be obtained.

AVAS-QUER writes the data either to an output file in SQL format, or to output files containing the data in LOAD or CSV format. LOAD is a special Informix format (please see the documentation for your Informix database). CSV (Comma Separated Value) is a format which can be processed by many database systems (e.g. by Microsoft Access).

Users must transfer the output file(s) created by AVAS-QUER to their target system. They then execute the files or import the data to the database system.

On the target system, the user can employ suitable database queries to obtain derived information such as the following:

1. In which nets is a particular job used?
2. In which nets is a particular symbolic date (symdat) used?
3. On which days is a particular symbolic date set?
4. Dependency structures in the form:
 - Which net depends on another specific net?
 - Which net depends on the contents of a particular job variable?
 - Which net depends on a job in another net?
 - Which net tests a particular resource?
 - Which net depends on a particular value?
5. In which objects is a particular document used?
6. Which jobs are running on a particular pubset, or which job variable contains the pubset?
7. Which user groups use a particular calendar?
8. In which orders is a particular SELECT-TURNUS used?
9. For which orders or nets is there a document in the DOCLIB?

BS2000 multiprocessor operation

The multiprocessor system HIPLEX MSCF allows different BS2000 business servers to be interconnected to form a computer network.

In AVAS multiprocessor operation, one particular processor is assigned the role of AVAS master. This processor runs the AVAS system including the run control system. AVAS can distribute entire nets or individual jobs within nets to any processor in the MSCF network and monitor the processing of these distributed jobs.

Thanks to the distributed handling of nets and jobs in multiprocessor operation, it is possible to ensure optimum load distribution while automatically taking into consideration any logical or temporal dependencies.

Authorization concept

The entire job production is protected by personalized access control via AVAS user IDs and passwords. Users can only invoke those AVAS functions which are entered in their function table.

A number of users can be combined to form a user group. Each user can only access the net and job libraries assigned to his or her user group. This allows the nets and jobs of different departments to be made mutually inaccessible.

Batch interface

Recurring tasks within the framework of AVAS control can also be handled on the procedural level. This is made possible by batch statements for the majority of the AVAS functions.

Program interface

A program interface provides read access to the run control and journal file, and thus to AVAS operative data. In addition, selected AVAS functions are access via a program interface.

User interface

The AVAS dialog system provides users with a mask-driven interface, which is implemented in BS2000 by FHS.

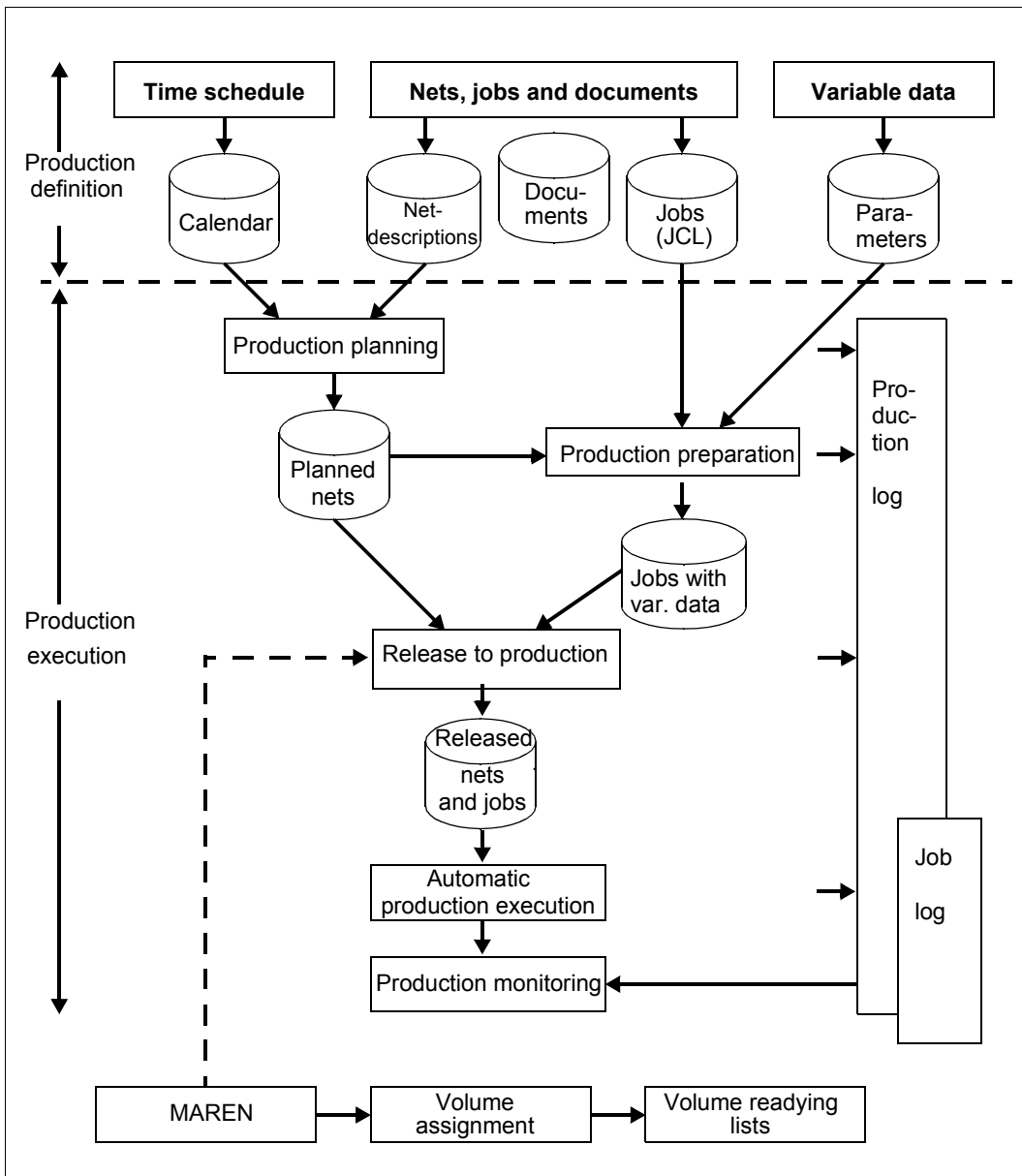


Figure 1: Schematic representation of the interaction of AVAS functions during production definition and execution

3 Production definition

Under AVAS, jobs are combined to job nets for production. These jobs are BS2000 jobs. Conditions can be used to control the nets. These only allow a net to continue running if a condition is fulfilled. Conditions can be set on the termination of other nets, jobs, resources and values defined by the user and on whether a specific value has been reached in a BS2000 job variable. Nets of this type are referred to as standard nets in AVAS.

AVAS allow nets to be started and monitored within other nets. Nets of this type are referred to as hypernets and the nets which are called from a hypernet are referred to as subnets. Nets cannot be started from within subnets.

Planning and starting a net on the basis of date and time specifications is implemented using symdats (symbolic date), which are entered in the AVAS calendar. The periods over which nets are to be planned and released must be defined. AVAS offers standard names for symdats and periods, such as WD for a working day or nxtweek for the period covering next week.

3.1 Nets for production run

The definition of the net specifies the start time at which a net is to be executed, the sequence in which the jobs are to be started and what conditions are to be fulfilled for execution to continue.

3.1.1 Net description

When a net is described, entries are required regarding the net, the jobs and the conditions.

Net specifications

Net name (NET-NAME)

When assigning net names, it is possible for all statements to be used with names truncated from the right. If, for instance, the nets of a procedure are to be displayed later, the net names of the nets in this procedure should be classified in descending order from left to right.

Documentation element (NET-DOC)

Name of a documentation element in a library. User-defined information on a net can be stored under this name and displayed in subsequent functions.

Selection criterion for net variants (SELECT-TURNUS)

SELECT-TURNUS can be used to create runtime variants of a net. Note, however, that the criterion cannot be altered if processing of two or more nets is scheduled via the calendar. Nets for which the SELECT-TURNUS value specified here is to be modified must be scheduled singly (selected with 'S' when planning takes place via the calendar).

Entry for the run control system (RUN-CONTROL-SYSTEM)

If, at AVAS system generation time, two or more run control systems were set up, the net can be assigned to one of these run control systems.

Entries for the /ENTER-JOB command to start the jobs and for the /ENTER-PROCEDURE command for starting the S procedures (SDF-P)

The following entries are required in the ENTER call to supply parameters to the start command concerned (the /ENTER-JOB command for starting jobs and the /ENTER-PROCEDURE command for starting S procedures). The job description specifies whether the parameters are taken from the net description and job description or from the LOGON statement of the job. Entries in the job description overwrite the values specified for the net.

- User ID (NET-USER)
- Account number (NET-ACCOUNT)
- Password (NET-PASSWORD)
- Job class (NET-CLASS)
- Job runtime log (NET-LOG)
- Catalog ID of slave processor (NET-CAT)
- Server name for a BS2000 job on a remote BS2000 system (NET-CAT)
- Further attributes for the selected job class and further parameters for the ENTER call concerned (NET-PARAMETER)

Net mask specifications

List of masks for net modification (FORMAT-NAME).

File with parameters for modification

The name of the parameter file containing the update values can be permanently stored in the net description. Moreover, a default mechanism can be used for assigning the parameter file.

If the default mechanism applies, AVAS looks for the file name PARAM.\$ug.netname[.date[.time]]. Valid ranges of parameter files result from partial/full qualification of the file name with date and time. AVAS first looks for the fully-qualified file name (including date and time). If no fully-qualified file is listed in the catalog, AVAS looks for the partial qualification with the date, and in a third step for the net name only. It is thus possible to specify a one-day or an unlimited validity range for a particular net. AVAS takes the date and time from the net name of the net occurrence created with CREATE-PLAN-NET. This enables the user to employ a procedure chain to supply the parameter files holding the update values to AVAS for automatic processing.

Job specifications

Job name (JOB-NAME)

As far as the abbreviation of names is concerned, the same rules apply to the assignment of job names as are valid for the assignment of net names. The job names may be structured using periods. Within AVAS, partial qualification using the period (.) is not permitted; the underscore character (_) must be used for this purpose.

Documentation element (JOB-DOC)

Name of a documentation element in a library. User-defined information (e.g. restart information) on a job can be stored under this name and displayed in subsequent functions.

Entry specifying where the ENTER parameters are taken from (ENTER-PARAMS)

This parameter controls whether the parameters for the ENTER call are to be taken from the SET-LOGON-PARAMETERS command (or LOGON) of the job or determined from the entries regarding the job or net in the net description. Permissible values for ENTER-PARAMS:

- LOGON
The ENTER parameters are taken from the SET-LOGON-PARAMETERS command.
- NET
The ENTER parameters are taken from the net description, with precedence being given to the job description entries.

Name of an external job intended to run under AVAS (ENTER-FILE)

This parameter need only be specified if JOB-TYPE=EXT was specified. If the ENTER-FILE has been given a password, this password must be specified under FILE-PASSWORD.

Password of an external job intended to run under AVAS (FILE-PASSWORD)

This parameter need only be specified if JOB-TYPE=EXT was specified and ENTER-FILE has been assigned a password.

FT request specifications

Request name (FT-NAME)

Like job name

Documentation element (FT-DOC)

Like JOB-DOC

Entries for the TRANSFER-FILE command

From the values of these parameters the run control system creates the TRANSFER-FILE command which is used to execute file transfer.

- **DIRECTION**
Transfer direction
- **PARTNER-NAME, REMOTE**
Name and operating system type of the remote host
- **LOCAL-FILE, REMOTE-FILE**
Names of the local and remote files
- **REMOTE-TRANSFER-ADMISSION**
FTAC authorization profile on the remote host
- **FT-PARAMETER**
Entry of further operands of the TRANSFER-FILE command for which no AVAS parameters are available (e.g. for the follow-up processing for the local or remote system). The syntax of the TRANSFER-FILE command must be adhered to. AVAS does not check the syntax.

Condition specifications

Documentation element (COND-DOC)

Name of a documentation element in a library. User-defined information on a condition can be stored under this name and displayed in subsequent functions.

Name of the condition (CONDITION-NAME)

Depending on the COND-TYPE parameter, here you should specify either:

- the name of the net or job whose predefined status is to be waited for
- the name of a condition description or job variable whose value is to be checked.

In the case of the JVA condition the job variable with the specified name COND-JVA-NAME must exist. The value of the job variable is then compared with the specified value. The following can be used as comparison operators here: =, >, <, >=, <= and <> (not equal to).

The value of the job variable is checked against the value range defined by JVA-POSITION and JVA-LENGTH. If the job variable is protected by a password, this password must be specified under JVA-PASSWORD.

The event check begins with the start of each index level.

3.1.2 Creating and editing nets

The CREATE-NET-DESCRIPTION statement enables the user to create a net description and store it as a net in the net library (NETLIB).

The MODIFY-NET-DESCRIPTION statement is used to modify a net.

The processing sequence and the requirements for starting the individual processing steps are defined in the network structure. In AVAS, index levels are used to arrange the jobs within a network.

All jobs on the same index level are started simultaneously. The various index levels are processed sequentially, i.e. the jobs on one index level are started only after all the jobs on the previously index level have been completed successfully.

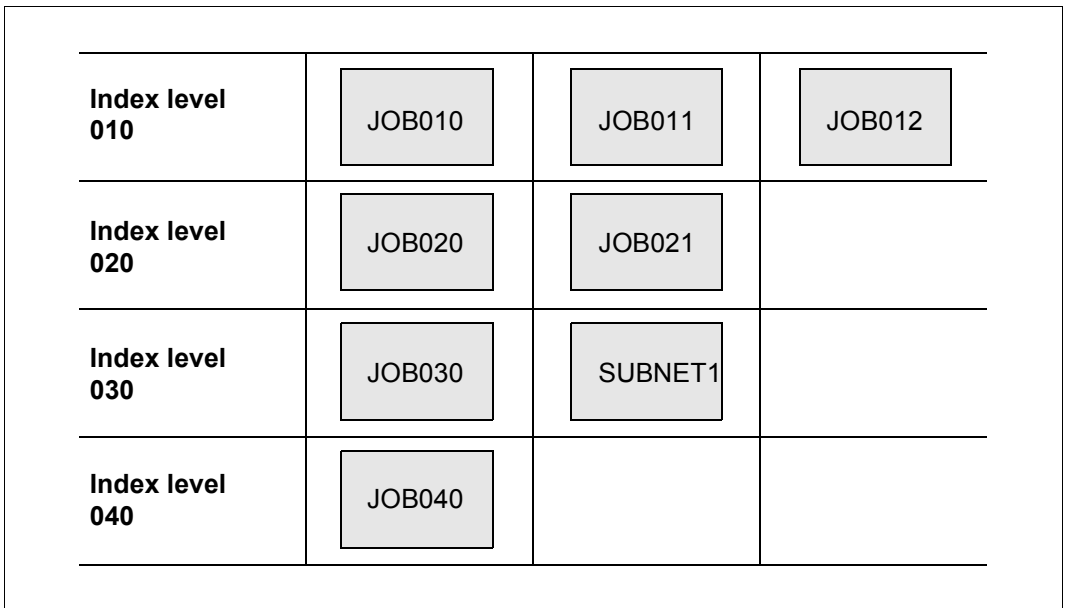


Figure 2: Example of an AVAS network with four index levels

In this example, JOB020 and JOB021 are started only after JOB010, JOB011 and JPB012 have been completed with no errors.

On index level 030 the subnet SUBNET1 is started in parallel to job JOB030.

To take advantage of existing parallelism, you can also use a job's synchronization index to specify the level on which the system must wait for the job to be completed.

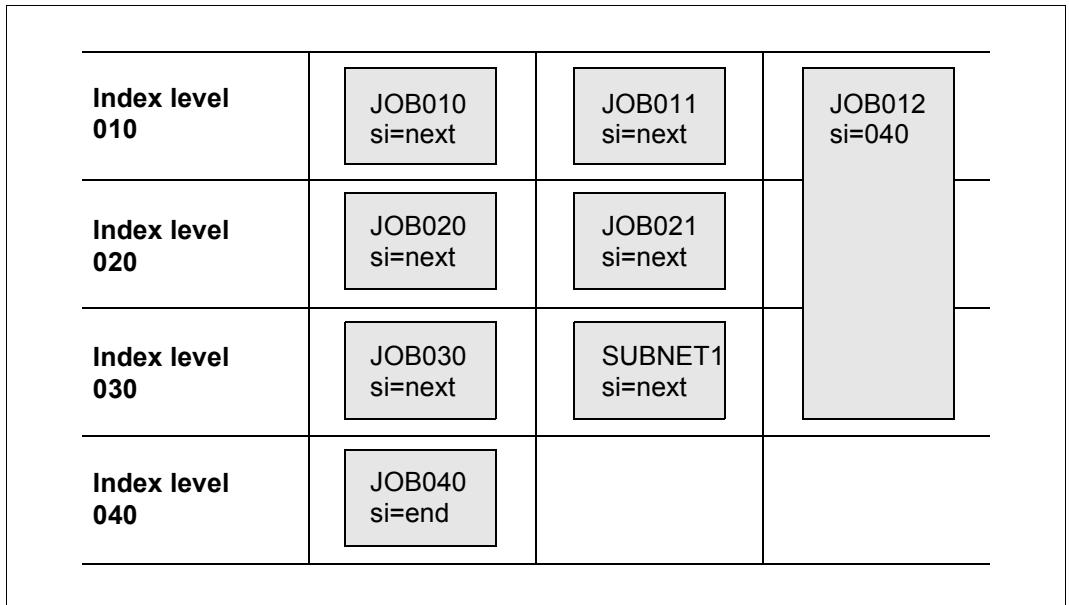


Figure 3: Example of an AVAS network with synchronization index

Because of the specification “synchronization index si=040”, JOB012 can run parallel to all jobs, with the exception of jobs of the final index level 040. All other jobs are started in stages according to the default value “si=next”.

Within the network structure, dependencies are specified in the same way as jobs.

Internally, a net description consists of structure elements. These structure elements must be defined for:

- each BS2000 job
- each S procedure
- subnets
- conditions governing the start of an index level

For the net, data must be defined for the symbolic and real start times, the net parameters, the default values for the job parameters, and the net masks required for modification purposes.

For the structure elements, data must be defined for selecting the structure element (TURNUS and SYMDAT), as must the assigned parameters, depending on the function of the structure element.

With the condition descriptions, data must be defined for condition selection (TURNUS, SYMDAT), the condition description (COND-NAME) and the condition value (COND-VALUE).

As a rule, net descriptions have the following format:

- Description of the net parameters (mask AVN001)
- Description of the net plan data (mask AVN020)
- Format specifications for user masks (mask AVN006)
- Net structure description (mask AVN004) with structure elements
 - for starting a BS2000 job or an S procedure
 - for starting an FT request
 - for starting a subnet
 - for defining a condition description
 - for modifying a condition description
 - for testing a condition
 - for deleting a condition description

All structure elements in a net must be assigned an index, in ascending order according to the order of net execution. The index sequence may contain gaps.

INDEX

```

001
:   Structure elements for starting jobs or procedures
:   Structure elements for starting subnets
:   and
:   for processing and testing
:   condition descriptions and conditions
899

900
:   Structure elements which must be executed
:   in addition during a restart after an error
:   Structure elements for starting subnets are not permitted
999

```

Within a net, a job may run more than once on different index levels. However, it may only be called once on a given index level.

In a net, the same name must not be used for a BS2000 job and an S procedure, otherwise the CHECK function records an error. The net cannot be planned in this case.

Subnets can only be planned once within a hypernet.

If the start of the first job in a net is to be made dependent on external conditions, only these conditions may appear on the lowest index level of the net. External conditions include:

- waiting for another net to terminate;
- waiting for the end of a job in another net;
- waiting for the availability of a resource;
- waiting until a VALUE type condition description has a preset value;
- waiting until a job variable contains a preset value.

Statements for net editing

The following statements are available for net editing. These statements are described in the “AVAS Statements” manual [1]:

CHANGE-NET-DESCRIPTION	Large-scale change to nets
COPY-NET-DESCRIPTION	Copy a net description
CREATE-NET-DESCRIPTION	Set up a net description
DELETE-NET-DESCRIPTION	Delete a net description
MODIFY-NET-DESCRIPTION	Modify a net description
SHOW-NET-DESCRIPTION	Display a net description

3.1.3 Defining BS2000 jobs, S procedures, FT requests and subnets

A structure element for starting a BS2000 job, an S procedure, an FT request or a subnet is described in the net description with FUNCTION=J, FUNCTION=P, FUNCTION=F or FUNCTION=S. The following types can be defined in the net description:

MOD

- FUNCTION=J/P

The BS2000 job or S procedure is saved in the AVAS system and contains variables which must be modified for every processing run.

It must be created as a temporary task by a CREATE-PROD-NET statement.

STD

- FUNCTION=J/P

The BS2000 job or S procedure is saved in the AVAS system and need not be modified for every processing run.

The production task must be created by a CREATE-PROD-JOB statement.

EXT

- FUNCTION=J/P

The BS2000 job or S procedure is not saved in the AVAS system and is started under the AVAS net control system by an /ENTER-JOB or /ENTER-PROCEDURE command. The BS2000 job or S procedure is assigned via the file name specified under ENTER-FILE (mask AVN002).

In the case of external BS2000 jobs and S procedures, NET must be specified in the ENTER-PARAM parameter field (mask AVN002).

EXX

- FUNCTION=P

The S procedure is not saved in the AVAS system and is started under the AVAS net control system by an /ENTER-PROCEDURE command. The S procedure is assigned via the file name specified under ENTER-FILE (mask AVN002). The contents of JOB-PARAMETER (mask AVN002) are assigned to the procedure parameters.

The monitoring job variable is not assigned to the S procedure as a task job variable (external S procedure with external monitoring job variable). The run control system sets up the job variable with an internal name.

The job variable is assigned a password according to the start parameters of the run control system.

The user must give the job variable a value just like with a task job variable. The contents are governed by the rules for BS2000 task job variables and for the AVAS statement #AVJ# (setting errors and restart variants via the task job variable).

If the SIGNAL and TRANSFER programs are to run under the external control system, you must make sure that these programs can address the job variable via the JV link name *SMONJVJ.

```

ENTER-PROCEDURE
  FROM-FILE=<ENTER-FILE>
  ,PROC-PASS=<FILE-PASSWORD>
  ,MONJV=*NONE
  ,PROCEDURE-PARAMETERS=(
    CONTROL-JVA-NAME=control-jva-name
    ,CONTROL-JVA-PASSW=control-jva-passw
    <Contents of JOB-PARAMETER>)

```

In the case of external S procedures, NET must be specified in the ENTER-PARAM parameter field (mask AVN002).

TRA

– FUNCTION=F

The FT request for file transfer to remote hosts is defined fully by the entries in the structure element.

These entries are not modified when net production takes place.

NET

– FUNCTION=S

The subnet is stored as a standard net in the library of the net descriptions. It becomes a subnet by defining it in a hypernet.

3.1.4 Time specifications in nets

Time specifications for starting a net

PLAN-START can be used to predefine real or symbolic start dates for the net. The use of different symbolic start dates (SYMDAT-NAME) is only meaningful if runtime variants of the net are formed via SYMDAT-NAME (use of SYMDAT-NAME for the job).

Specifying a real date as the starting point also enables nets to be planned in a cycle.

At the “production planning” stage, the real start date of the net is derived from PLAN-START. It becomes part of the net name in the production plan and can no longer be modified.

Moreover, an earliest start date EARLIEST-START is also introduced for the net.

This date can be changed when the net is planned or released. (At first, EARLIEST-START is identical to PLAN-START, being part of the net name.)

The same applies to a subnet planned with the hypernet via a structure element with FU=S/TYPE=NET. PLAN-START is taken from the hypernet if no symdat with a PLAN-START value is entered in the subnet.

If a symdat with the same name as the one used to plan the hypernet is entered in a subnet, PLAN-START is taken from the hypernet.

NET-TYPE makes it possible to serialize nets with the same name but different start times (PLAN-START).

- NET-TYPE 1: The net is started when EARLIEST-START has been reached. Any number of nets with the same name can run concurrently.
- NET-TYPE 2: The net is not started if a net of the same name but with a different start time is in progress. If a queue of nets with the same name arises, these are processed in the sequence defined by PLAN-START.
- NET-TYPE 3: The net is only started if no net with the same name has been executed since the last time the job file was reorganized (the entry in the contents of the job file is critical).

The same applies for subnets, except that NET-TYPE 5, 6, and 7 are entered for these (corresponding to 1, 2 and 3).

An AVAS net will only run if the following conditions are satisfied:

- the earliest start time EARLIEST-START is reached
- the latest possible start time LATEST-START has not yet been exceeded. LATEST-START time refers to the planned start time of the net (PLAN-START).
- the net is of type 1 or type 2 (NET-TYPE) and no other net of the same name (or plan start) is running, or else it is of type 3 and no net of the same name has been run since the last reorganization.

If several nets of type 2 or 3 with the same name are ready to run, the one with the oldest PLAN-START time is started first. The restrictions for these nets apply only within the same run control system.

The parameter DELAY-SOLUTION describes what happens to a net that has exceeded its LATEST-START time

- WAIT The net goes on waiting
- START The net is started nevertheless
- IGNORE The net is not started, dependencies of other nets are deemed to be resolved
- CANCEL The net is not started and is deemed to have been canceled because it was errored.

The PLAN-START parameters are taken from the hypernet if no symdat is entered in the subnet. If a symdat with the same name as the one used to plan the hypernet is entered in a subnet, the PLAN-START parameters are taken from the hypernet. The following are affected by these parameters:

- Nets with NET-TYPE=2 or 3, if a net of the same name prevented execution beyond LATEST-START
- Nets for which release with SUBMIT-NET is delayed
- Nets which were put into the status HOLD prior to the start and were only passed for processing with RESUME-NET after the latest start time as defined by LATEST-START and
- Nets which were not started because the run control system was not active between PLAN-START and LATEST-START.

A subnet that was planned together with the hypernet with the same start time is started when the corresponding structure element FU=S/TYPE=NET is activated. The state of the subnet is changed from NETWAIT (where it waits after the net has been released) to WAITING, by which the subnet is passed to the control of the run control system.

Symdats with a preceding “!” can be used for subnets (see [page 46](#)). If the subnet was planned using a !symdat with its own start time, it is started at this time by the run control system.

It is possible to manually start a subnet, even if the structure element with FU=S/TYPE=NET has still not reached the time at which it should be activated.

Conditions for processing a structure element

A net consists of three types of elements or tasks, the jobs, conditions and sub nets. The latest time by which processing must have begun can be specified for jobs and conditions: the LATEST-START time for jobs or the LATEST-OCCURE time for conditions. The LATEST-START and LATEST-OCCURE times refer to the planned start time in the net name (PLAN-START).

The following conditions must therefore be satisfied for a structure element to be processed:

- The net must run and must not be stopped
- The LATEST-START or LATEST-OCCURE time must not be exceeded
- All structure elements whose synchronization index is less than or equal to the index of the element must be successfully processed.

If a structure element is not processed because its start time has been exceeded, further execution of the net depends on its DELAY-SOLUTION parameter:

- **START** The task is started despite the time being exceeded, and the condition is regarded as satisfied.
- **IGNORE** The task is not started, but further processing of the net continues as after a normal termination, and the condition is regarded as satisfied.
- **CANCEL** The task is not started and processing of the net is canceled because of errors.

In the case of subnets, these checks are carried out using the parameters of the subnet itself if DELAY-SOLUTION = START is entered for the structure element with FU=S/TYPE=NET.

3.1.5 Defining conditions in net descriptions

It is possible to test the following types of conditions in a net description:

NET condition	Satisfaction of a condition on another net
JOB condition	Satisfaction of a condition on a job or FT request
RESOURCE condition	Satisfaction of a condition on a resource
VALUE condition	Satisfaction of a condition by a defined value
JVA condition	Satisfaction of a condition by a defined value of a job variable
TIM condition	Satisfaction of a condition with a specific wait time

The RESOURCE, VALUE and TIM conditions can be defined in the net description. As a result they can be used in the network itself and in other networks.

NET condition/JOB condition

A NET condition is used to wait for a condition on another net to be satisfied. The other net may be in the user's own group or another one.

A JOB condition is used to wait for a condition on a job to be satisfied. The job may belong to the same net or to another one.

Conditions from FT requests are treated like conditions of jobs; they consequently belong to the type JOB condition.

Nets and jobs referred to in such dependencies must be identified in the net description via the LIFE-TIME parameter.

When either a SUBMIT or REPEAT is executed for the net, a condition description is then initialized in the run control file, containing the data for the net or job as applicable. Subsequently, this entry will be given the appropriate status (e.g. the entry is set to the appropriate status (ENDED, ERROR,...) after the net or job terminates). If the net or job is omitted, or if the job is deleted, the values IGNORED or DELETED will be set.

A condition definition has the lifetime entered under LIFE-TIME in the net description.

If dependencies on other nets and jobs are defined in a net, a check will be made at the time of their execution to determine whether a condition description (already) exists:

- If there is already a condition description for the condition (e.g. in the case when a condition on another net is tested), details of the net which is waiting are noted in the condition.
- If, on the other hand, no description exists yet, the status NO-OCCURE will be set for the condition.

The descriptions for NET and JOB conditions can be edited in a dialog using the statements DELETE-COND-DESCRIPTION, MODIFY-COND-DESCRIPTION and SHOW-COND-DESCRIPTION.

- DELETE-COND-DESCRIPTION enables a condition description for a NET/JOB condition to be deleted.
- MODIFY-COND-DESCRIPTION enables the status and the LIFE-TIME of the condition description to be modified.
- SHOW-COND-DESCRIPTION allows the user to display the nets which are waiting for a NET/JOB condition to be satisfied, and secondly to output the contents of a condition description (net or job creating it, status of the condition, status of the entry).

In testing a NET or JOB condition, the test will be applied to the net or job which has a PLAN-START before and closest to the PLAN-START of the dependent net.

RESOURCE condition

A RESOURCE condition is defined, modified, deleted and tested within the net structure.

During definition and testing, a resource can be allocated as exclusive (EXCLUSIVE) or shared (SHARE). In AVAS, a resource is a logical term defined by a user. The user (alone) associates the term to real BS2000 resources (e.g. files).

The maximum number of users who can use a resource defined to be allocated in SHARE mode is specified using the MAX-USING-SHARE parameter. The value range is 2 through 100. If no value is specified, MAX-USING-SHARE=100.

When a structure element with FUNCTION=C and TYPE=RES tests the condition, the resource is only allocated in SHARE mode if at least as many allocation units are available as are defined in OCCURE-VALUE= SHARE(uu). uu specifies the number of allocation units the user wishes to allocate.

A resource within the same net can be allocated several times with different values for SHARE(uu) using FUNCTION=C with TYPE=RES.

The status OCCURRED is set in the structure element when the resource is allocated. The status NO-OCCURE is set when fewer allocation units are available than are to be allocated with SHARE(uu) or if the resource is in the status CREATED, EXCLUSIVE or ERROR.

A resource can be released again with the modify function (FU=M).

If a single allocation of a resource (COND-VALUE=FREE and SHARE(uu)) is released for a structure element with FU=M, TYPE=RES and COND-VALUE=FREE, the allocation value (USING) is set to zero.

If a net has allocated a resource via several structure elements with FU=C and TYPE=RES several times SHARE(uu), the entry with the oldest timestamp is located and deleted. This also applies if the resource was allocated with FU=A with TYPE=RES using COND-VALUE=SHARE(uu). Partial release of allocation using FREE(uu) is not permitted.

A structure element with FU=M and TYPE=RES must be assigned to each structure element with FU=C and TYPE=RES in the net to ensure that the resource is no longer allocated by the net when the net has completed execution. The resource is not released explicitly when the net terminates. If allocation by a net was not released, the administrator can use MODIFY-COND-DESCR to delete such allocation.

A resource can be deleted again with the delete function (FU=D). Deletion only takes place if the resource is not in use and no net is waiting to allocate the resource. If the resource cannot be deleted, the status ERROR is set for the structure element.

If a RESOURCE condition is created during the processing of a net, a corresponding condition description is noted in the run control file. If a RESOURCE condition is tested and

- there is not yet a condition description in the run control file, the status NO-OCCURE will be set for the condition.
- there is a condition description in the run control file, the testing of the RESOURCE condition will produce the following results, depending on the assignment status:
 - a resource with COND-STATUS = FREE will be allocated in SHARE or EXCLUSIVE mode as appropriate.
 - a resource with COND-STATUS = SHARE will be assigned to the additional net in SHARE mode (even if there is an outstanding request for allocation in EXCLUSIVE mode).
 - If a resource is allocated the number of times defined in MAX-USING-SHARE, further requests are entered in the WAITING queue.
 - If a resource is allocated in EXCLUSIVE mode, it cannot be allocated with SHARE.
 - The request is entered in the queue.
 - Allocation in with SHARE mode is carried out according to FIRST-IN. This does not apply if, for instance 10 resource allocation units are available and FIRST-IN is waiting for 20 allocation units and a subsequent user for 10 units or less. If allocation is possible, it is always implemented.
 - If several nets are waiting for the exclusive assignment of a resource, then when this resource next has the status FREE it will be assigned to the net which was first to request its allocation.

The net which tests the condition is noted, with the time of first testing, in the condition description.

The descriptions for RESOURCE conditions can be edited in a dialog using the statements ADD-COND-DESCRIPTION, MODIFY-COND-DESCRIPTION, DELETE-COND-DESCRIPTION and SHOW-COND-DESCRIPTION.

- ADD-COND-DESCRIPTION enables a RESOURCE condition to be defined, even outside a net.
- MODIFY-COND-DESCRIPTION enables a RESOURCE condition to be released. It is only possible to modify MAX-USING-SHARE if the resource is not allocated and if no net is waiting to allocate it. The USING value cannot be modified for SHARE.
- DELETE-COND-DESCRIPTION enables a condition description for a RESOURCE condition to be deleted. It is only possible to delete the condition description if the resource is not allocated and if no net is waiting to allocate it.
- SHOW-COND-DESCRIPTION allows the user to display the nets which are waiting for a particular status, and thus a condition to be satisfied, and secondly to output the contents of a condition description (net or user creating it, status of the condition, status of the entry).

VALUE condition

A VALUE condition is defined, modified, deleted and tested in the net structure.

Modification can be used to set a different value. When a VALUE condition is tested, the relationships “equal to”, “not equal to”, “greater than”, “greater than or equal to”, “less than”, “less than or equal to” can be tested. It is also possible to combine two values using a logical “OR”.

If a VALUE condition is created during the processing of a net, a corresponding condition description is noted in the run control file. If a VALUE condition is tested and

- there is not yet a condition description in the run control file, the status NO-OCCURE will be set for the condition.
- there is a condition description in the run control file with a different value, the testing net will be noted in this condition description as waiting.

The descriptions for VALUE conditions can be edited in a dialog using the statements ADD-COND-DESCRIPTION, MODIFY-COND-DESCRIPTION, DELETE-COND-DESCRIPTION and SHOW-COND-DESCRIPTION:

- ADD-COND-DESCRIPTION enables a VALUE condition to be defined, even outside a net.
- MODIFY-COND-DESCRIPTION enables a VALUE condition to be modified.
- DELETE-COND-DESCRIPTION enables a condition description for a VALUE condition to be deleted.
- SHOW-COND-DESCRIPTION allows the user to display the nets which are waiting for a particular value, and thus a VALUE condition to be satisfied, and secondly to output the contents of a condition description (net or user creating it, status of the condition, status of the entry).

Condition control

For each of the condition types, NET, JOB, VALUE and RESOURCE, the user can prescribe a value or the OCCURE-VALUE status for which the condition is considered to be satisfied. It is also possible to specify a list of status values for OCCURE-VALUE. In addition, the user can specify an ERROR-VALUE for any condition, which will cause the condition itself to have an ERROR status.

As in the case of jobs, it is possible to set three restart variants for the conditions. One restart variant can be directly assigned to the ERROR-VALUE (this corresponds to the variant specified by the MONJV in the case of a job). This means that it is also possible in the case of conditions to initiate a prescribed restart.

JVA conditions

A JVA condition waits for the condition to be satisfied by a job variable taking a defined value. In this case the value of the job variable is compared with the specified value. Possible comparison operators are: =, <, >, <=, >= and <> (not equal to).

No condition descriptions are created for the JVA condition and it is not possible to specify OCCURE-VALUE and ERROR-VALUE.

TIM condition

To enable a system wait until a prescribed time to be effected at any required point in the net structure, a new structure element has been introduced with the function "WAIT for TIME". This enables the user to specify an earliest start point for jobs, or the earliest time at which a condition is to be tested.

Other timing controls for conditions

It is possible to specify LATEST-OCCURE and OCCURE-DELAY-SOLUTION parameters for any of the conditions.

LATEST-OCCURE can specify either a relative time (relative to PLAN-START) or an absolute time. The behavior when this time point is passed is controlled by the specification of OCCURE-DELAY-SOLUTION. The possible values of OCCURE-DELAY-SOLUTION are START, IGNORE or CANCEL. Specifying these leads to the condition status OCCURRED, IGNORED or ERROR.

This enables a condition to be treated as omitted or errored; in the case of the condition status ERROR, net processing must be resumed by a restart.

Statements for editing

The following statements can be used for editing condition descriptions of the types NET, JOB, RESOURCE and VALUE.

ADD-COND-DESCRIPTION	Add a condition description
DELETE-COND-DESCRIPTION	Delete a condition description
MODIFY-COND-DESCRIPTION	Modify a condition description
SHOW-COND-DESCRIPTION	Display a condition description

No condition descriptions are created for conditions of type JVA.

3.1.6 Defining and editing hypernets and subnets

A hypernet is a net in which at least one structure element with FUNCTION=S and TYPE=NET is used to plan, modify, release and monitor a net. A net such as this which is called from a structure element is referred to as a subnet. The structure element also shows the status of the relevant subnet (PLANNED, CREATED, WAITING, RUNNING, ENDED etc.).

The definition of structure elements with FUNCTION=S is not permitted within a subnet. A check is carried out when the hypernet is planned.

The subnet is started by starting the structure element with FUNCTION=S and TYPE=NET.

A separate start time can be defined for subnets:

- If a subnet is to be started at a later time after the structure element with FU=S and TYPE=NET has been activated, the planned start symdat of the hypernet must be entered in the subnet prefixed by "!". The start time and the start parameters from this !symdat are used.
- If a subnet is to be started at the time the structure element with FU=S and TYPE=NET is activated, the symdat used to plan the hypernet must be removed from the subnet.

Subnets should only contain symdats with "!" in order to achieve greater transparency.

Symdats with a preceding "!" are not used for net planning when planning is performed via the calendar.

Index level 010	FU=J JOB010 TYPE=STD	FU=J JOB011 TYPE=MOD	FU=S Net012 TYPE=NET
Index level 020	FU=C \$Bk.NET020 TYPE=NET	FU=S Net021 TYPE=STD	
Index level 030	FU=C JOB030 TYPE=JOB	FU=S Net031 TYPE=NET	FU=J JOB032 TYPE=MOD
Index level 040	FU=J JOB040 TYPE=EXT		

Figure 4: Example of a hypernet structure with subnets

3.2 Jobs for the production run

This section describes the creation and acceptance of jobs and the editing of jobs and JCL elements. It also discusses how jobs are supplied with current execution parameters via the USER-PARAM-FILE and user masks.

Note

FT requests are defined fully by the structure element. They therefore contain no JCL data and are consequently not affected by the information in this section.

3.2.1 Creating and editing jobs

Jobs and JCL elements are created or edited by means of the EDIT-JOB statement. Jobs, which have already been created and are executable in BS2000 or in a server system can be included in the AVAS environment (see the COPY-ELEMENT statement).

AVAS distinguishes the following job types:

- BS2000 jobs
They are started by AVAS using ENTER-JOB.
- S procedures
Procedures which are created according to the SDF-P rules. They are started by AVAS using ENTER-PROCEDURE.

AVAS statements and variables can be incorporated into any jobs and JCL elements. They enable the user to encode in the individual job or S procedure, those values which vary from run to run, thereby making them easier to modify. Within the framework of job modification, these encoded values are then converted with the aid of user masks and the USER-PARAM-FILE into current values and incorporated into the jobs and JCL elements. In the case of S procedures, the encoded values are assigned the current values and are stored in the parameter area.

The SIGNAL and TRANSFER programs can be integrated into BS2000 jobs or S procedures to enable runtime logs to be transferred to a central library administered by AVAS. The logs are made known to AVAS by the SIGNAL program and transferred to the central library by the TRANSFER program.

The jobs are stored in the job library JCLLIB. They can be made accessible to all users by copying them to the system job library JCLSYS (COPY-SYSTEM-ELEMENT statement).

Once created, the jobs are combined to form a job net, known for short as a net. The properties of the net must be set down in the net structure description. As well as the jobs, this also contains dependencies as structure elements. A net structure is created using the CREATE-NET-DESCRIPTION statement. It is stored in the net library NETLIB. Once stored, a net structure can be modified by means of the MODIFY-NET-DESCRIPTION statement or displayed with the SHOW-NET-DESCRIPTION statement. The net description can be made accessible to all users by copying it to the system net library NETSYS (COPY-SYSTEM-ELEMENT statement).

Groups of nets and jobs can be combined to form a hypernet. In the structure of the net, a job can be described which causes a subnet to be started (structure element FU=S).

Documentation elements can be assigned via the net description:

- to the net as a whole
- to the individual structure elements of the net.

The documentation elements are stored in the documentation library DOCLIB. The documentation elements can be made accessible to all users by copying them to the system documentation library DOCSYS (see the COPY-SYSTEM-ELEMENT statement).

The AVAS-QUER utility is supplied to facilitate maintenance of net data. This reads the basic AVAS data in BS2000 and selects data for further processing in relational databases. Appropriate database queries allow relevant information to be derived (see “AVAS-QUER” on [page 20](#)).

3.2.2 AVAS statements and variables in jobs and JCL elements

Jobs and JCL elements are created and edited in the user library JCLLIB by means of the EDIT-JOB statement.

When jobs and JCL elements are created or edited, it is possible to incorporate additional information into the JCL statements. This is done by predefining symbolic names, which are interpreted by the AVAS system as AVAS statements and AVAS variables. They can only be used in the following contexts:

- facilities for adapting values which vary from run to run
- facilities for assigning JCL elements which are stored externally
- facilities for passing run control information to AVAS with the aid of a task job variable
- restart facilities following errored behavior on the part of a task.

AVAS statements are executed within the framework of modification. The run control information is evaluated by the run control system.

The following symbolic names, when used in jobs and JCL elements, are automatically reserved by the AVAS system and are interpreted as statements. They must be located in the task starting at column 1 of a record.

Symbolic name	Function of the statement
#AVM#	Assign a mask for run parameter input.
#AVS#	Call a JCL element.
#AVD#	Call an element from any library or file.
#AVJ#	Send information to AVAS using a task job variable.
#AVA#	Enter job information in the journal.
#RA	In the event of a restart, replace the next statement by the contents of this statement.
#RI	In the event of a restart, insert the contents of this statement.
#RU	In the event of a restart, suppress the next statement.

AVAS statements can also be entered as /REMARK or /WRITE-TEXT commands. In this case, they must begin in column 9 or column 15, respectively:

```
/REMARK #AVM#...
/WRITE-TEXT C'#AVM#...
```

Jobs containing AVAS statements in /REMARK or /WRITE-TEXT commands can be executed under BS2000 without being modified. The function of the AVAS statements, however, is only carried out at the production preparation stage within AVAS (CREATE-PROD-NET/CREATE-PROD-JOB). AVAS recognizes the symbolic names of the statements, even if preceded by /REMARK or /WRITE-TEXT commands, and performs the corresponding function.

The following symbolic names are recognized as AVAS variables. They may be located in any record, starting at any location within that record.

Symbolic name	Function of the variables
P#nnn	AVAS variables from job masks (CREATE-PROD-JOB/CREATE-PROD-NET).
S#nnn	AVAS variables from system fields (nnn = 000–200) The user's AVAS variables (nnn = 201–999) (CREATE-PROD-JOB/CREATE-PROD-NET).
N#nnn	AVAS variables from net masks (COLLECT-NET-PARAMS, CREATE-PROD-NET).
F#nnn	AVAS variables from the USER-PARAM-FILE (CREATE-PROD-JOB/CREATE-PROD-NET).

The symbolic name indicators P#, S#, N# and F#, which are normally preset for the AVAS system, can be defined differently by the AVAS administrator. Internal processing is however optimized on the use of the standard indicators.

3.2.3 Creating the AVAS user masks

User masks can be used to supply jobs with current runtime parameters. To do this, the jobs must be given dummy parameters, which are replaced by values from the user masks. In this way, the variables in a job can be modified at the production preparation stage with the current values required for each run, i.e. they can be replaced and assigned to the job.

User masks can be created using FHS or IFG. AVAS supports +-formats and #-formats in the user masks.

There are two kinds of user mask:

- masks for entering parameters which are valid throughout the net, i.e. net parameters (COLLECT-NET-PARAMS statement) and
- masks for entering parameters for a particular job (CREATE-PROD-NET/CREATE-PROD-JOB statements), which are called from the job via an AVAS statement (#AVM#).

As regards the information, control and message sections, every user mask must have the same structure as a system mask.

Lines 2–21 contain the application-specific processing section.

The control section must contain the variable fields for the operation, statement and operands, with the text field names and the sequence of entries being the same as in the system masks. Interchanging these mask fields will cause processing errors.

The same remarks apply to the message section as to the control section.

If a user mask does not comply with the prescribed AVAS conventions, it is not possible to perform parameter modification via masks. The mask is rejected by AVAS with an error message.

The user is responsible for assigning the variable fields in the masks to the variables in the jobs and JCL elements. When creating masks, he should make sure that these variables are not used more than once per mask, and remember that only the most recently entered values apply during subsequent processing (no memory function). This means that a mask must contain **all** the information required for the subsequent processing operation.

Values are assigned using the FHS parameters EXIT and REM; the name of the AVAS variable must be specified under REM.

When IFG is used, EXIT=YES and REM=name must be specified for the variable fields, where “name” is the name of the AVAS variable.

3.2.4 Editing jobs and JCL elements

Under AVAS, jobs and JCL elements which are stored in the AVAS user library JCLLIB can be edited using the AVAS statement EDIT-JOB.

External jobs and S procedures, i.e. those outside the AVAS environment, can only be edited using the EDT statement. They can be copied to the JCLLIB using the AVAS statement COPY-ELEMENT.

External server jobs can be edited on their respective systems using an editor, or they must be imported onto the AVAS host and copied to the JCLLIB with the COPY-ELEMENT statement.

Statements for editing jobs and JCL elements

Elements in the JCLLIB can be edited with the statements below, which are described in the “AVAS Statements” [2] manual.

EDIT-JOB	Edit jobs and JCL elements (with EDT)
EDT	Edit external SAM/ISAM user files (with EDT)
DELETE-JOB	Delete jobs and JCL elements
SHOW-JOB	Display jobs and JCL elements (via EDT)

3.2.5 Editing documentation elements

Descriptions can be created for nets and individual structure elements.

Documentation elements are stored in the documentation library DOCLIB. The elements are displayed and edited using EDT.

For this purpose, EDT is called such that the documentation can be displayed and entered in uppercase and lowercase letters.

If the ELEMENT-NAME operand is specified in the statements, either one element or a directory of all the elements from the documentation library DOCLIB is displayed. The form of partial qualification specified for the element name determines what is displayed. Individual elements can be selected for editing with mark S or Y.

Editing of documentation elements by way of the ELEMENT-NAME operand has no effect either on the assignment of the documentation elements to the nets or to the specifications in the net.

The documentation elements are assigned to the condition descriptions in the ADD-COND-DESCRIPTION and MODIFY-COND-DESCRIPTION statements.

The documentation elements are assigned to the nets and to the structure elements of the nets with the CREATE-NET-DESCRIPTION and MODIFY-NET-DESCRIPTION statements.

Elements from the system documentation library DOCSYS can be displayed by means of the SHOW-DOCUMENT statement if the element name is specified with \$ugsys.

Statements for editing documentation elements

Documentation elements can be edited, displayed and deleted with the following statements (see the “AVAS Statements” manual [1]):

DELETE-DOCUMENT	Delete documentation elements
EDIT-DOCUMENT	Edit documentation elements
SHOW-DOCUMENT	Display documentation elements

3.3 Restarts in nets

AVAS provides options for restarting in the event of abnormal termination of net runs. Such restarts can be carried out automatically or manually after the error has been eliminated. Errored jobs can be repeated, skipped or replaced by a different job.

With RESTART, FT requests are treated like jobs.

3.3.1 Preparing for the restart

When processing of a net has been abandoned because of an error or a CANCEL-NET statement with CANCEL-TYPE=SOFT (the net has the status ERROR), the net can be restarted through a RESTART-NET statement. Parameters must be specified in the net description for this eventuality. These parameters must take account of the fact that nets with a status of RUNNING can be processed by structure elements that have the status ERROR.

If a structure element in a net has terminated with an error without causing processing of the net itself to be abandoned (the net has the status CALLED FOR ERROR), the error status for the net can also be reset by the RESTART-NET statement.

Up to three restart variants can be allocated to each structure element. They define how the net is to be resumed when the POINT-OF-ERROR for the net occurred in that structure element.

POINT-OF-ERROR

The POINT-OF-ERROR in the net structure is defined by the element that was allocated a status of ERROR as the result of an abnormal termination or by the statement CANCEL-NET.

In situations where several structure elements have been given a status of ERROR, every such element can be selected as the POINT-OF-ERROR via the structure display (mask AVD007).

When it comes to processing a structure element in the POINT-OF-ERROR, one of the three restart variants specified for that element can be selected.

A restart variant consists of the following parameters:

- **RESTART-INDEX**
Index in the net at which processing is to be resumed.
- **RESTART-NAME**
Parameter for selecting the structure element of the **RESTART-INDEX**
- **RESTART-TYPE**
Processing mode for the structure elements in the **RESTART-INDEX**
- **AUTOMATIC**
Type of restart

A restart variant is deemed to be valid if it causes the status of the structure element at the **POINT-OF-ERROR** to change to **WAITING** or **SKIPPED** and the required changes in status of the elements are valid. Valid status changes can be specified in the options (see [page 64](#)).

If an element still has a status of **ERROR** following a restart, another restart for this element can be issued immediately or later.

POINT-OF-RESTART

The **POINT-OF-RESTART** is defined by that **RESTART-INDEX** specified for the selected restart variant. Processing of the net following a restart resumes with this index, with **RESTART-INDEX=END** forming an exception. No **POINT-OF-RESTART** is defined in this case.

In cases where the restart index has a number of structure elements assigned to it, the **RESTART-NAME** parameter can be used to make the required selection. Furthermore, when the restart is initiated by a **RESTART-NET** statement, individual elements can be excluded from processing by marking them accordingly in mask **ADV005**. Which structure elements are to be processed on the subsequent index levels is determined by the net structure and the synchronization index.

The status is not changed if a structure element is excluded from the processing.

If the **POINT-OF-RESTART** lies within the range of the restart index levels (900–999), all elements linked via the **SYNC-INDEX**, up to the first index of normal processing (index 001–899), are displayed in the **AVD005** mask.

From the elements displayed on the first index level of normal processing, those which are to be processed on restart can be selected.

No selection is possible if the element in the **POINT-OF-ERROR** lies within the range of the restart index levels. Making a selection (**RESTART-NAME** parameter and entering a mark in mask **AVD005**) at the restart index level means that processing of the element in the **POINT-OF-ERROR** cannot be prevented.

Restart index

RESTART-INDEX is the name given to an index in the net at which point processing can be resumed. A normal processing index (index 001–899) or an index that is only processed at a restart (index 900–999) can be specified as the restart index level. RESTART-INDEX=END is also a valid specification for structure elements in the range of normal processing.

The specified restart index level must ensure that the ERROR status of the element at the POINT-OF-ERROR changes to a SKIPPED or WAITING status following the restart. To ensure that the element at the POINT-OF-ERROR is processed, the index level specified as the RESTART-INDEX must be associated with the element at the POINT-OF-ERROR. This is done through the synchronization index.

This means that when a restart is performed through index level 9nn, the index for the return to normal processing (index 001–899) must be in the synchronization index of the element at the POINT-OF-ERROR. If the element at the POINT-OF-ERROR is a restart element (index levels 900–999), the restart index level must address the processing sequence that has a RESTART-INDEX greater than index 899. The first index level of normal processing with RESTART-NAME=*ALL (net repetition) and the index for the return to normal processing (RESTART-INDEX and RESTART-NAME from the restart variant 1 of the restart element in which the return is defined) is also permitted.

Restart types

The type of restart, and consequently the processing mode for the structure element in the POINT-OF-RESTART, is defined by the RESTART-TYPE and AUTOMATIC parameters of the selected restart variant:

RESTART-TYPE=RESTART

The structure element or elements in the POINT-OF-RESTART is/are modified by the run control system via the AVAS restart statements #RA, #RI and #RU before the start.

If a server job is restarted with RESTART-TYPE=RESTART, this means:
Before the AVAS agent AVSSINCM is started, the run control system returns the contents of the server area (positions 129–256) of the last run to the task job variable. AVSSINCM recognizes from the contents whether the job has to be restarted (start mode of AVSSINCM) or whether the status of the server job is to be obtained (QUERY mode of AVSSINCM).

RESTART-TYPE=NORMAL

The structure element or elements in the POINT-OF-RESTART is/are brought to execution without being modified by the AVAS restart statements.

If a server job is restarted with RESTART-TYPE=NORMAL, this means: AVSSINCM is restarted and branches to start mode. The contents of the job variables (of the last run) are deleted.

A multiple restart at a restart index level means that the mode specified for an element may be superseded by another mode. The following procedure has been adopted to cater for this eventuality:

RESTART mode overwrites NORMAL mode. By contrast, NORMAL mode does not overwrite an already defined RESTART mode.

If the restart index level for the sequence linked through the synchronization index jumps forward (old RESTART-INDEX → new RESTART-INDEX) as the result of a new restart, the mode of all the relevant elements at the old restart index level will be deleted.

The mode of restart jobs in the range of index levels 900–999 can be switched from RESTART mode to NORMAL mode or from NORMAL mode to RESTART mode through the RESTART-TYPE parameter of restart variant 1.

Restart variant 1 must also be used here to define the return to index levels 001–899. RESTART mode is exited automatically when the first index level of normal processing has been completed.

AUTOMATIC=YES

The restart is normally initiated automatically by the run control system, without requiring entry of the RESTART-NET statement. If a restart is initiated through a restart variant with AUTOMATIC=YES, AUTOMATIC is reset to NO (only one automatic restart can be performed through a restart variant).

The user only needs to initiate the restart with RESTART-NET if

- the structure element could not be started due to incorrect ENTER parameters,
- processing of the nets has been interrupted by the run control system with the /INFORM-PROGRAM MSG=' STOP , LEVEL=NAME' , JOB-ID=*TSN(<tsn>) command, or
- processing of the net has been aborted by the CANCEL-NET statement with CANCEL-TYPE=SOFT.

AUTOMATIC=NO

The restart must be initiated through the RESTART-NET statement.

Modifications can be made

- to the net, using MODIFY-SUBMIT-NET and/or
- to structure elements of the net, using MODIFY-SUBMIT-JOB.

The MODIFY-SUBMIT-NET/MODIFY-SUBMIT-JOB statements are not selected automatically.

Using the RESTART-NAME parameter to select structure elements on the restart index level**RESTART-NAME=name**

Of the structure elements specified in the index level in the RESTART-INDEX, only this one will be executed again. Structure elements which had not yet been processed up to this point (WAITING) cannot in this case be excluded. The RESTART-NAME parameter can only be used to affect the RESTART-TYPE (see also "[Status changes in the POINT-OF-RESTART](#)" on page 61).

If a structure element is to be selected using FU=W and TYPE=TIM, the name must be specified as *DATE. The specified RESTART-NAME must enable the structure element to be uniquely identified.

If the specified element is not in the restart index level, the restart will have to be performed from the AVD005 mask.

RESTART-NAME=*ALL

All structure elements in the specified index level in the RESTART-INDEX are executed again.

RESTART-NAME=*ERROR

All structure elements in the specified index level in the RESTART-INDEX that terminated abnormally (STATUS=ERROR) are executed again. The *ERROR parameter is only processed when the specified index level in the RESTART-INDEX is the same as the index of the element in the POINT-OF-ERROR.

RESTART-NAME=*NAME

Only the structure element in the POINT-OF-ERROR is executed again.

The *NAME parameter is only processed when the specified index level in the RESTART-INDEX is the same as the index of the element in the POINT-OF-ERROR.

The functions CREATE-NET-DESCRIPTION, MODIFY-NET-DESCRIPTION and MODIFY-SUBMIT-NET supply the RESTART-NAME operand with various default values.

- If RESTART-INDEX is identical with the structure element index, RESTART-NAME will by default be supplied with the values from the DEFAULT-RESTART-JOB generation parameter.
- If RESTART-INDEX is greater or smaller than the structure element index, RESTART-NAME will be default with *ALL.

In cases where the restart is taking place at index levels 900–999, normal processing must be resumed at the index of the POINT-OF-ERROR if RESTART-NAME=*NAME, or if *ERROR is specified in restart variant 1 of the restart element in which the return was defined.

The marks in the POINT-OF-RESTART that were set using the RESTART-NAME parameter of the selected restart variants of the element in the POINT-OF-ERROR can be modified by the operator before initiating the restart (mask AVD005).

The marks set cannot be modified if the POINT-OF-ERROR is in the range of restart index levels.

Processing of the structure element in the POINT-OF-ERROR must not be prevented by the choice of restart index level.

The RESTART-NAME parameter is ignored if RESTART-INDEX=END is specified in the restart variant.

Selecting a restart variant

The priority restart variant is selected by specifying the task job variable in the job with the AVAS statement #AVJ#RV=n.

The restart variant set via the task job variable is entered by the run control system in the structure element for the job in the POINT-OF-ERROR. The restart is initiated via this restart variant unless the user specifies another restart variant via mask AVD012, AVD007 or AVD005.

In the case of FT requests it is not possible to use the AVAS statement #AVJ#RV=n.

If a restart variant is defined neither via the task job variable nor via the structure element description nor by the user via mask AVD012, it will be requested via the parameter field of mask AVD007 or AVD005.

The restart variant specified via the task job variable or in the structure element is displayed in a special field in mask AVD007 (parameter field R-V).

Status changes in the POINT-OF-RESTART

If there are two or more structure elements on the index level of the POINT-OF-RESTART, the following general rules apply to the status changes resulting from the RESTART-NET function:

M	IND	Status before restart	Status after restart	Remark
Y	080	ERROR	WAITING	No status change in branch (WAITING)
	080	ERROR	ERROR	No status change in branch (WAITING)
Y	080	ENDED	WAITING	All structure elements in the branch are set to WAITING
Y	080	SKIPPED	WAITING	All structure elements in the branch are set to WAITING
	080	SKIPPED	SKIPPED	No status change in the branch
Y	080	WAITING	WAITING	No status change in the branch
	080	WAITING	WAITING	No status change in the branch
	080	ENDED	ENDED	No status change in the branch
	080	RUNNING	RUNNING	Job must not be selected
	080	NO-OCCURE	NO-OCCURE	No status change in the branch (WAITING)
	080	OCCURRED	OCCURRED	No status change in the branch
*Y	080	OCCURRED	WAITING	All structure elements in the branch are set to WAITING
	080	EXECUTED	EXECUTED	No status change in the branch
Y	080	EXECUTED	EXECUTED	All structure elements in the branch are set to WAITING
Y	080	EXECUTED	WAITING	Only if FU=M, TYPE=RES is the EXECUTED status reset to WAITING if RESTART-WAIT-CONDITION =YES is specified (see page 65)

M Denotes the mark column. On execution of the RESTART-NET statement, the marks are set internally in accordance with specifications made via the RESTART-NAME in the POINT-OF-ERROR (see CREATE-NET-DESCRIPTION statement).

IND Denotes the index of the POINT-OF-RESTART. The value 080 is selected here as an example.

Status before restart

Denotes the STATUS displayed with the RESTART-NET statement prior to execution of the function.

Status after restart

Denotes the STATUS set by the RESTART-NET statement after execution of the function.

- * Note relating to the condition in OCCURRED status
The change in status of the condition depends on the optional setting (see [page 64](#)).

A branch is defined by the structure elements linked via the SYNC-INDEX.

Status changes of jobs and conditions

If the status of the structure element in the POINT-OF-ERROR is not modified by the selected restart variant (changed to SKIPPED or WAITING), the restart is rejected with an ERROR message or result.

The same applies if an element that has a status of RUNNING is specified in the selected restart variant.

POINT-OF-RESTART equals POINT-OF-ERROR

In principle, the only structure elements affected by the restart are those linked through the SYNC-INDEX to selected elements at the restart index level.

All elements with a status not equal to WAITING are given a status of WAITING. If an element with a status of ERROR (but not the element at the POINT-OF-ERROR) has to be given a status of WAITING, the RESTART-WAIT-ERROR system parameter can be used to specify whether a restart is to be permitted or not.

The conditions in NO-OCCURE status are not processed; they remain in NO-OCCURE status.

Conditions with a status of OCCURRED are handled by the RESTART-WAIT-CONDITION system parameter in accordance with the optional settings.

POINT-OF-RESTART precedes POINT-OF-ERROR

All structure elements selected in the POINT-OF-RESTART and all the subsequent elements linked to them via the SYNC-INDEX will be processed. Stand-alone structure elements (those without a predecessor; their INDEX is not used as a SYNC-INDEX) and their branches will only be processed when their index is started by the run control system with the same priority as an index in the SYNC-INDEX sequence that is to be processed.

The status of each element is set to WAITING if it is not WAITING already.

The conditions in NO-OCCURE status are not processed; they retain the NO-OCCURE status.

Conditions with the status OCCURRED are handled by the RESTART-WAIT-CONDITION system parameter in accordance with the optional settings.

If an element with the status ERROR (but not the element at the POINT-OF-ERROR) has to be given a status of WAITING, the RESTART-WAIT-ERROR system parameter can be used to specify whether a restart is to be permitted or not.

POINT-OF-RESTART follows POINT-OF-ERROR

All that is processed during a restart is the branch of the net between POINT-OF-ERROR and POINT-OF-RESTART (including the structure elements linked to this branch through the SYNC-INDEX). All the elements in the branch, including the element at the POINT-OF-ERROR, are given the status SKIPPED if their status is not ENDED or OCCURRED.

All other branches of the net with SYNC-INDEX=RESTART-INDEX are not affected by the restart, and the structure elements retain their current status (they are not given the status SKIPPED).

If a job with the status ERROR (but not the element at the POINT-OF-ERROR) has to be given the status SKIPPED, the RESTART-SKIP-ERROR system parameter can be used to specify whether a restart is to be permitted or not.

If a condition with the status NO-OCCURE has to be given the status SKIPPED, the RESTART-SKIP-CONDITION system parameter can be used to specify whether a restart is to be permitted or not.

RESTART-INDEX is in the range of restart jobs

The restart jobs of the selected processing sequence on the index levels 900–999 are given the status WAITING.

The return index (RESTART-INDEX restart variant 1) is interpreted as a RESTART-INDEX (Index level 001–899).

How the net structure is processed (change in status) depends on the relationship between the return index to the index job at the POINT-OF-ERROR.

RESTART-INDEX=END

In all branches of the job at the POINT-OF-ERROR, the jobs with a status that is not ENDED and the conditions with a status that is not OCCURRED are given the status SKIPPED. The branches are processed as far as their end condition (SYNC-INDEX= END).

If the job at the POINT-OF-ERROR itself has SYNC-INDEX=END, only one job will be given the status SKIPPED.

If a job with a status of ERROR (but not the job at the POINT-OF-ERROR) has to be given the status SKIPPED, the RESTART-SKIP-ERROR system parameter can be used to specify whether a restart is to be permitted or not.

If a condition with a status of NO-OCCURE has to be given the status SKIPPED, the RESTART-SKIP-CONDITION system parameter can be used to specify whether a restart is to be permitted or not.

Optional settings

Certain default values in accordance with the requirements of the computer center can be defined for restart processing. These settings are defined via the system parameters and are applicable throughout the system. The AVAS administrator can thus determine whether or not the RESTART-NET operation is to be used to effect the required change in status.

The default values RESTART-SKIP-ERROR and RESTART-SKIP-CONDITION describe how the RESTART-NET operation responds when the choice of restart variant means that a part of the net is not to be executed.

RESTART-SKIP-ERROR={YES / NO}

YES Structure elements with a status of ERROR can be allocated a status of SKIPPED.

NO The restart will be rejected if a structure element status has to be changed from ERROR to SKIPPED.
The only exception to this is the structure element at the POINT-OF-ERROR.

RESTART-SKIP-CONDITION={YES/NO}

YES Conditions with a status of NO-OCCURE or WAITING can be allocated the status SKIPPED.

NO The restart will be rejected if a condition status has to be changed from NO-OCCURE or WAITING to SKIPPED.

The default values RESTART-WAIT-ERROR and RESTART-WAIT-CONDITION describe how the RESTART-NET operation responds when the choice of restart variant means that a part of the net is to be executed again.

RESTART-WAIT-ERROR={YES / NO}

YES Structure elements with a status of ERROR can be allocated the status WAITING.

NO The restart will be rejected if a structure element status has to be changed from ERROR to WAITING.
The only exception to this is the structure element at the POINT-OF-ERROR.

RESTART-WAIT-CONDITION={YES / NO}

YES Conditions with a status of OCCURRED are to be changed to the status WAITING. They are retested following the restart to see whether the condition is satisfied.

Structure elements with FU=M, TYPE=RES and the EXECUTED status are also set to the WAITING status because FU=C, TYPE=RES reserves a resource and FU=M, TYPE=RES releases it again.

Structure elements with FU=D and FU=A, TYPE=RES are not changed from EXECUTED to WAITING.

After processing of a structure element with FU=D, TYPE=RES, the resource can no longer be addressed after a restart.

NO Conditions with the status OCCURRED are not processed during the restart. They retain their OCCURRED status and are not checked following the restart.

Note that resources are no longer reserved after processing with FU=M, TYPE=RES.

The resource may have to be reserved again via an index of the restart processing with FU=C, TYPE=RES.

Conditions whose status has changed from NO-OCCURE to SKIPPED as a result of the RESTART-NET statement are deemed to be satisfied and are treated by RESTART-NET as conditions with the status OCCURRED.

All status changes (JOB-STATUS, COND-STATUS and NET-STATUS) are logged in the net journal.

If the user performs several partial restarts (AVD007 mask; several structure elements with a status of ERROR processed), a journal record is output for each one.

If the net cannot be restarted because of an error (ERROR result displayed in the AVD012 mask, an error message is sent to the net journal.

3.3.2 Restarts in hypernets and subnets

A structure element for starting a subnet is put in the status RUN/ERROR if the status ERROR is set for a structure element in the subnet and no further job is running in the subnet. In this event a restart must be initiated for the subnet.

No special rules apply to the restart variants in the subnet.

If the subnet contains no further structure element with the status ERROR, the additional ERROR status in the hypernet is also deleted.

The following rules apply to restart variants for structure elements with FU=S/TYPE=NET for starting subnets:

- It is only possible to specify RESTART-INDEX=ERROR-INDEX, RESTART-NAME=*NAME and AUTOMATIC=NO if processing of the subnet is still to be monitored by the hypernet.
- If RESTART-INDEX>ERROR-INDEX is specified for a restart variant, the status SKIPPED is set for the structure element in the hypernet and processing of the subnet is no longer monitored by the hypernet. In this event, no restart is initiated for the subnet. It must then be put into the status ENDED or ABENDED explicitly to allow the subnet to be deleted from the job and journal files during reorganization.
- A structure element with FU=S for starting a subnet is given the status ERROR when:
 - The subnet was canceled with CANCEL-NET or
 - When the structure element is started, the structure element and thus the subnet is regarded as having terminated abnormally since the value CANCEL has been set for NET-DELAY-SOLUTION in the event of a delay.
- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, the status ERROR is set for the structure element with and the subnet is not started if
 - the subnet is not in the status NETWAIT or ENDED because, for instance, the subnet was started manually before it was initiated by the structure element with FU=S and is still in the status RUNNING.
 - NET-TYPE>4 is not set for the subnet because, for instance NET-TYPE is set to a value <4 in order to edit the subnet with MODIFY-SUBMIT-NET.
 - The subnet is not in the job file because, for instance, it was manually started and then terminated and the AVS.REORG procedure has already been deleted from the job file.
 - the subnet cannot be read, for instance because it is locked (RESULT=LOCKED)
- If the subnet is in the status ERROR, this has the following consequences:
 - No AVAS-SUBNET job variable is set for the subnet.
 - The subnet cannot be started and edited via the hypernet.
 - The subnet must be put into the status ENDED or ABENDED by the user.

- If after a manually initiated start (for NET-TYPE < 4), the subnet reaches the status ENDED, the status ENDED can be passed to the hypernet by executing the statement RESTART-NET with the parameter RESTART-INDEX=ERROR-INDEX for the hypernet. In all other cases, a restart must be performed for the hypernet with RESTART-NET and RESTART-INDEX>ERROR-INDEX in order to set the status SKIPPED for the structure element with FU=S and TYPE=NET.

Depending on the status a structure element has reached, this has the following consequences for restarting the corresponding subnet:

- If a structure element for starting a subnet has reached the status ENDED or SKIPPED in the subnet, the status is no longer set to WAITING by RESTART-NET. The subnet is no longer started by the hypernet.
- If a structure element for starting a subnet has reached the status ERROR in the hypernet because the subnet has reached the status ABENDED, the only restart variants which are possible are those in which the status SKIPPED is set.
- If a structure element with FU=S and TYPE=NET has reached the status ERROR in the hypernet because the subnet to be started does not have the status NETWAIT, RESTART-NET can be used to determine the result for the structure element when the subnet has terminated.
- If the subnet to be started is in the status WAITING, MODIFY-SUBMIT-NET can be used to reset the status NETWAIT using NET-TYPE>4, after which a restart can be initiated using RESTART-INDEX = ERROR-INDEX.

Statement for restarting nets in the event of an error

RESTART-NET Restart a net after an error

3.4 Time scheduling

Time scheduling in AVAS is done using calendars and period functions.

3.4.1 Calendars

All scheduling activities are performed using calendars that extend over several years and include a day column containing real dates and symbolic dates - the symdats assigned to individual calendar days. Certain days can be declared as non-working days (NWRK) or free days to distinguish them from working days. This forms the basis for typical exception rules such as WDPFRI='working day following (past) Friday that is not a working day'.

AVAS is supplied with a calendar that already contains all the usual important dates in symbolic form.

The calendar and the job nets are linked by means of symbolic start dates. In the net description, symbolic start dates can be assigned to the net as a whole or to individual jobs or dependencies in the net. During production planning, the calendar days are evaluated, and the symbolic dates entered are used to determine which nets are to be executed by this date.

In order to perform time scheduling via a calendar, each user group must have a calendar supplied with current dates. By default the name of the calendar is preset for each user group by means of the system parameters. A net can, however, also be assigned a special calendar in the net description.

Statements for editing calendars

The following statements are available for processing calendars. These statements are described in the "AVAS Statements" manual [1].

COPY-CALENDAR	Copy a calendar with its symdats
CREATE-CALENDAR	Set up a new calendar
DELETE-CALENDAR	Delete a calendar
MODIFY-CALENDAR	Modify a calendar
SHOW-CALENDAR	Display a calendar

Calendar 2005 2006	Days 01.01.2005	Symdats SAT NWRK NEW_YEAR
	01.12.2005	THU WD
	27.12.2005	TUEI WD
Period WEEK	28.12.2005 29.12.2005 30.12.2005 31.12.2005 01.01.2006 02.01.2006	WED WD THU WD FRI WD ULTIMO SAT NWRK NY's_EVE SUN NWRK NEW_YEAR MON WD WDNSUN
	03.01.2006	TUE WD WDPTUE
	31.01.2006	TUE WD ULTIMO
	31.12.2006	SUN NWRK NY's_EVE

Figure 5: Example for a calendar with symbolic dates (symdats) and a period

3.4.2 Periods

Periods can be defined in order to facilitate long-term scheduling for processing. The periods define a freely selectable term within the calendar.

All standard periods are supplied, e.g. TODAY, (current) WEEK etc. They are updated automatically by AVAS on a daily basis and can be named by the user.

Standard periods can be created using the batch statement CREATE-PERIOD (see the "AVAS Statements" manual [1]). The start and end date of standard periods is defined in a variable way and can therefore be changed along with the current date or the day of the week.

Standard periods are displayed using MODIFY-PERIOD, SHOW-PERIOD and DELETE-PERIOD with TYPE=VAR, but they cannot be changed or deleted.

Statements for editing periods

The following statements are supplied for editing periods. These statements are described in the "AVAS Statements" manual [1]:

CREATE-PERIOD	Create a period
DELETE-PERIOD	Delete a period
MODIFY-PERIOD	Modify a period
SHOW-PERIOD	Display a period

4 Production execution

This chapter describes production execution, which comprises the following steps:

- Production planning
- Production preparation
- Release for production
- Automatic production execution
- Production monitoring

It also includes a description of BS2000 multiprocessor operation with AVAS and how to control remote systems with AVAS-SV-BS2.

4.1 Production planning

During production planning, networks are added to the run. The following items are defined: The start times, the net variants (created on the basis of the Symdats) and the actions in the event of a delay (see [section “Time specifications in nets” on page 36](#)).

4.1.1 Planning standard nets

The freely selectable period for a forthcoming production run can be defined either using the calendar, or in the case of individual nets, without the calendar.

During production planning, AVAS nets are linked to real start dates, supplied with variable data for the actual production run, and incorporated in the production plan.

You can use symbolic dates to form subnets (referred to as net variants) from a net. A net element (job, condition) is only planned into the production run if the symbolic start date of the net is included in the list of symbolic dates of the element.

Thus, a **single** net can be used for different runs on different days, as illustrated by the following example:
 On Mondays and Fridays, Job011 is also run, and on Fridays Job032 is also run.

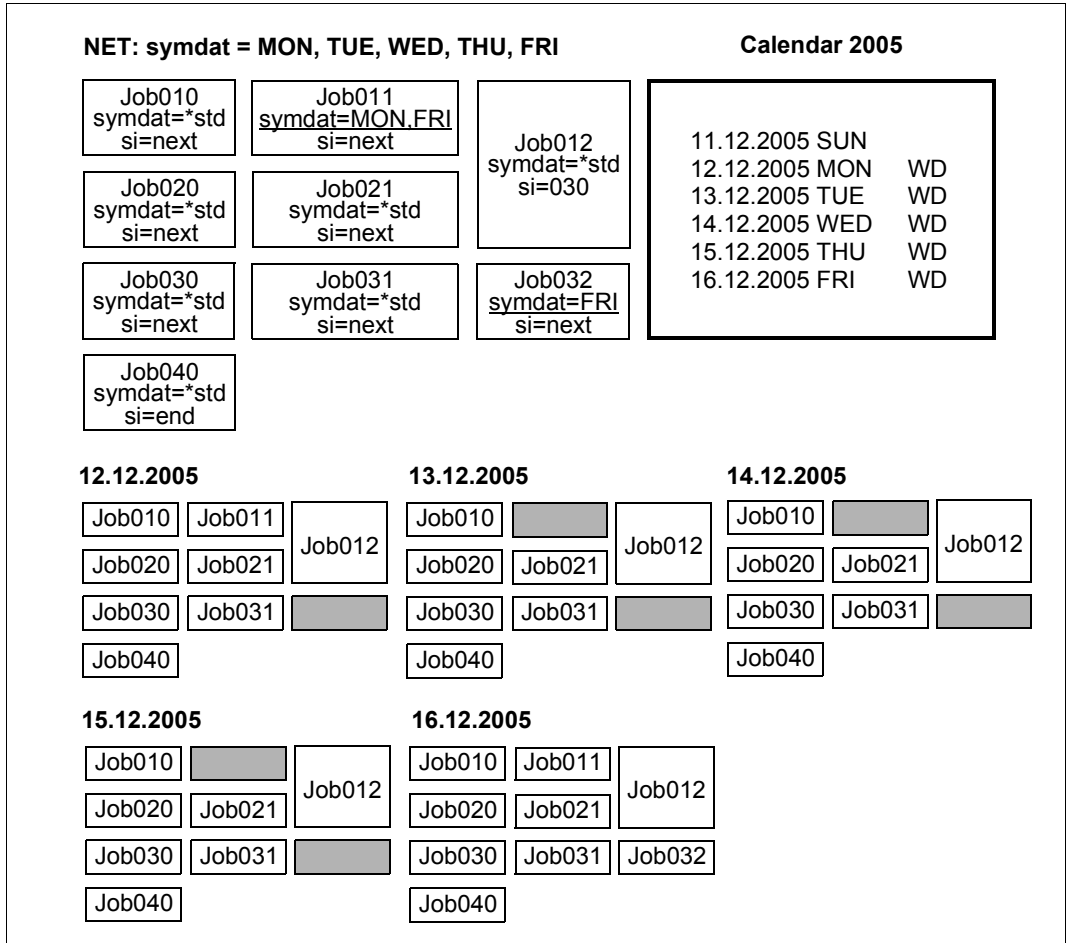


Figure 6: Example of production planning with net variants

From the AVAS production plan, it is possible to determine which nets are run on which days, and their current stage of preparation.

Statements for planning nets

The following statements are available for production planning. These statements are described in the “AVAS Statements” manual [1].

CREATE-ORDER	Plan, produce and release nets
CREATE-PLAN-NET	Plan net processing
DELETE-PLAN-NET	Delete planned nets and corresponding jobs in the production plan
MODIFY-PLAN-NET	Modify planned nets in the production plan
SHOW-PLAN-NET	Display the processing status of planned nets

4.1.2 Linking symdats

The symbolic dates (symdats) can be linked when the nets to be planned are selected and the net structure of the planned nets is formed.

Links are presented with the sign “+” or “-” ahead of the symdat names. In the masks a link is specified in the SYMDAT fields with a maximum length of 20 characters. Individual symdats can also be allocated a sign.

Net planning (selection of the nets to be planned)

In the case of net planning with a period beyond the calendar PLAN-START symdats can be allocated signs and linked. A distinction must be made between the following cases:

SYM1 Planning a net when SYM1 is entered in the calendar (planning without sign/link).

SYM1 + SYM2
Planning a net when SYM1 and SYM2 are entered in the calendar.

SYM1 – SYM2
Planning a net when SYM1 and not SYM2 is entered in the calendar.

-SYM1 Planning a net when SYM1 is not entered in the calendar.

Example

Planning a net for every day of the week except Thursday:

PLAN-START presetting = WT – THU

Net structure

The network structure must be ascertained for a planned network. This is done using the SELECT symdat. Here the SELECT symdat is dependent on the type of planning (with/without calendar) and on the PLAN-START symdat. Links with “+” and “-” are also possible with the symdats of the structure elements. The net structure is ascertained as follows:

Type of planning	PLAN-START symdat	SELECT symdat	Structure element with SYMDAT=	Selection
With calendar	<symdat1>	<symdat1>	<symdat1>	Yes
			*STD	Yes
			±<symdat2>	Check against calendar
	[±]<symdat3> [±]<symdat4>	*CAL	[±]<symdat3> [±]<symdat4>	Check against calendar
			*STD	Yes
Without calendar	*NONE	*NONE		Yes

Example

SELECT symdat	Structure element with SYMDAT=	Selection
SYM1	SYM1	Yes
	-SYM2	If SYM2 is not entered in the calendar
*CAL	WT + ULTIMO	To end of month
	WT-MON	Every work day except Monday
	SYM3	If SYM3 is entered in the calendar

Note

The linking of symdats or symdats with signs and the selection of structure elements with SELECT-SYMDAT was introduced with AVAS V7.0. If no links/signs are used, AVAS behaves as in the earlier versions.

4.1.3 Planning hypernets and subnets

This section describes special issues which must be taken into account when carrying out production planning for hypernets compared with standard nets.

Planning (CREATE-PLAN-NET), production preparation (CREATE-PROD-NET), and release (SUBMIT-NET) of the subnets is always carried out via the hypernet. The hypernet makes specifying the production sequence much easier than is possible using end-of-net conditions.

During planning $\text{NET-TYPE} = \text{NET-TYPE} + 4$ is set for subnets - subnets have NET-TYPE parameter values 5 through 7 (corresponding to 1 through 3 for standard nets). The hypernet controls processing of the subnets. Therefore, subnets are not shown in the net overviews in the DELETE-PLAN-NET, CREATE-PROD-NET, MODIFY-PROD-NET, DELETE-PROD-NET and SUBMIT-NET statements (except if the parameter DISPLAY=ALL is used in DELETE-PLAN-NET). In these statements, they are displayed and edited via the hypernet.

In all other statements, it is not possible to edit the subnets via the hypernet. Therefore, subnets appear in the net overview in all other statements and can be edited with the associated statements. You should note the following:

- Changes to the parameters or status of a subnet in the hypernet do not affect the subnet itself. The user is responsible for putting the subnet in the appropriate status.
- Changes to the subnet do not affect the status of the subnet in the structure element with FU=S and TYPE=NET in the hypernet and thus do not affect the status of the hypernet.
- MODIFY-PLAN-NET statement:
It is not possible to change $\text{NET-TYPE} > 4$ to $\text{NET-TYPE} < 4$ or $\text{NET-TYPE} < 4$ to $\text{NET-TYPE} > 4$. The subnet property cannot be removed or assigned.

Editing the subnets in the NPRLIB production plan

Executing the statement CREATE-PLAN-NET creates the subnets and the links to the hypernet. A structure element with FU=S and TYPE=NET is selected for a net variant of the hypernet if the value *STD is selected for Symdat or if the planning Symdat is entered.

If no planning symdat with “!” is found for a subnet planned in this way, the subnet is assigned the start time from the hypernet and the start parameters from the structure element with FU=S and TYPE=NET. The value for NET-DELAY-SOLUTION is set to START.

If the planning symdat with “!” is found for a subnet planned in this way, and if the start time of the subnet is not earlier than that of the hypernet (positive planned start difference), the start time and the start parameters for this subnet are used. In this case, the value for NET-DELAY-SOLUTION in the AVAS structure element FU=S is not set to START as a rule.

Example

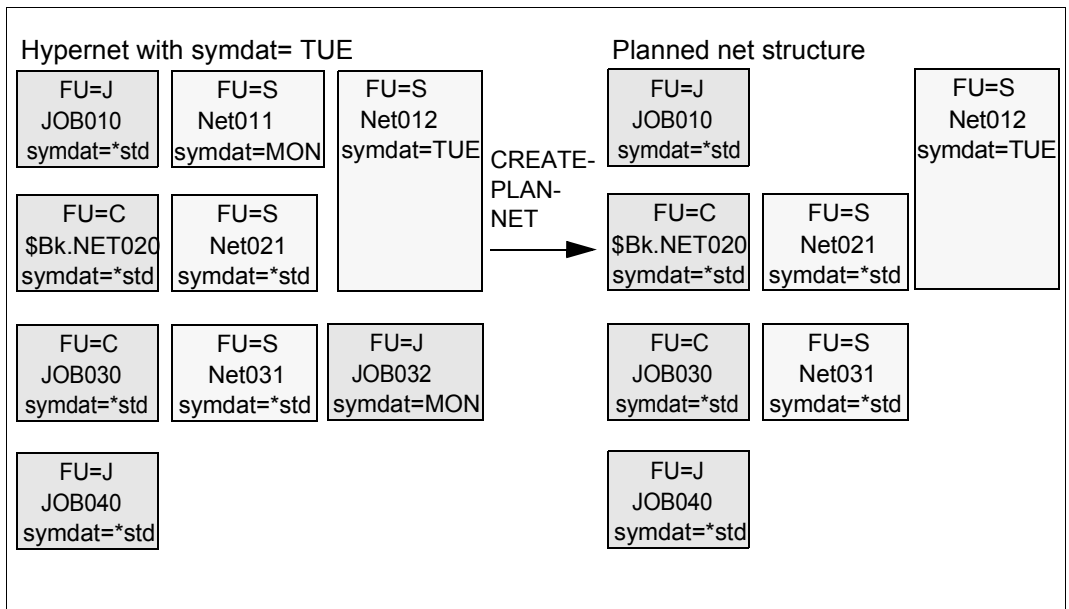


Figure 7: Example of planning a hypernet with subnets with CREATE-PLAN-NET

The subnets are edited via the hypernet. Planned subnets are deleted together with the hypernet with DELETE-PLAN-NET. They are only displayed in the net overview if DISPLAY=ALL is specified.

4.2 Production preparation

The main purpose of production preparation is to provide the current input data for planned jobs, i.e. to update values that vary from job to job (parameters).

These values are either AVAS or procedure parameters entered in the JCL. Escape characters define the positions at which parameters should be entered in the jobs.

AVAS offers the following options for assigning current values to net or job parameters:

- mask-driven assignment in a dialog
- values in parameter files
- user and system variables managed by AVAS such as Company, Date and Time

The options involving the use of a parameter file or variables managed by AVAS support fully automatic production preparation, e.g. where the user automatically supplies error-free, up-to-date values through upstream computerized processes.

In the case of manual assignment, the mask-driven dialog minimizes the risk of typing errors.

AVAS keeps preparation separate from production, i.e. preparation can complete the tasks assigned to it via AVAS asynchronously to production. This means that a net that is to run more than once can be supplied with the appropriate parameters for each separate date in advance.

Production preparation statements

The following statements are available for production preparation. These statements are described in the “AVAS Statements” manual [1].

COLLECT-NET-PARAMS	Collect parameters for modifying all tasks in a net.
CREATE-PROD-JOB	Create static BS2000 jobs and S procedures.
CREATE-PROD-NET	Create the temporary BS2000 jobs and S procedures of a net.
DELETE-PROD-JOB	Delete static tasks.
DELETE-PROD-NET	Delete all temporary tasks from a net.
EDIT-PROD-JOB	Edit executable BS2000 jobs and S procedures.
MODIFY-PROD-NET	Delete individual temporary tasks from a net.
SHOW-FORMAT	Display a user mask.
SHOW-PROD-JOB	Display executable tasks.

4.2.1 Modifying standard nets

For all jobs in a net, parameters which change from run to run are updated by means of production preparation at the parameter modification stage, with a distinction being made between net modification and job modification. A job may be either an ENTER job or an S procedure. FT requests are not subject to modification.

Net modification involves those jobs which are permanently assigned to a net; job modification involves standard jobs which can be brought to execution in any net.

During modification, the AVAS variables in the jobs and JCL elements are replaced by the values applicable to the run. The current values are assigned via

- user masks,
- a USER-PARAM-FILE,
- using the user's system variables S#nnn,
- from the system parameters or
- values defined by AVAS.

The length of the value is determined via the input medium (mask or user file).

Net modification

The net modification is performed with CREATE-PROD-NET. The executable nets are generated in the process.

The following prerequisites must be met:

- The net was added to the production plan at the "plan DP production" stage.
- The net has not yet been released.
- If modification is to be implemented via one or more masks, the masks must have been created beforehand by the user.
- The static jobs (jobs in the JMDLIB which can be assigned to more than one net) and JCL elements must have been created.

Net modification is performed in two stages and requires the following actions:

1. Stage: COLLECT-NET-PARAMS

This enables net parameters to be entered and modified via the net masks.

- The user masks specified in the net description are presented for parameter input.
- The parameters entered are stored in the net with their values and codes.
- The parameters stored for net modification may be displayed for modification any number of times via the net masks, provided no executable jobs have been created for the net by means of CREATE-PROD-NET.

2. Stage: CREATE-PROD-NET

This creates executable jobs which are permanently assigned to a net.

- The jobs marked JOB-TYPE=MOD in the net description are processed.
- The masks called in the jobs (#AVM#) are displayed (modification of P#nnn parameters).
- The JCL elements called in the jobs (#AVS#) are inserted in the JCL of the temporary jobs.
- The external elements called in the jobs (#AVD#) are inserted in the JCL of the temporary jobs.
- The net mask parameters stored for the net modification are processed and entered in the JCL (modification of N#nnn parameters).
- The assigned parameter file (SAM user file) is processed (modification of F#nnn parameters).
- The AVAS variables S#nnn are replaced by current values.
- The executable job is moved to the library of modified jobs (JMDLIB) as a temporary job (i.e. a job assigned uniquely to a net).
- The status CREATED is set in the production plan for the processed job.

The runtime data is entered in the net description:

- The status CREATED is set in the production plan for the processed net when all jobs to be modified in the net have been given the CREATED status.
If the jobs are not to be modified all at once, the net is given the PARTIALLY status, and is thus partially modified. If they are called again, jobs which were already created are not presented again for modification.

A log is created for the journal.

It is only possible to remodify a job in a net if the production job was deleted beforehand (MODIFY-PROD-NET).

Job modification

Job modification is performed by means of CREATE-PROD-JOB. This creates executable standard jobs.

- The masks called in the jobs (#AVM#) are displayed (modification of P#nnn parameters).
- The assigned parameter file is processed (modification of F#nnn parameters).
- The JCL elements called in the jobs (#AVS#) are inserted in the JCL of the standard jobs.
- The external elements called in the jobs (#AVD#) are inserted in the JCL of the temporary jobs.
- The AVAS variables S#nnn are replaced by current values (see the “AVAS Statements” manual [1]).
- The executable job is moved to the library of modified jobs (JMDLIB) as a static job (i.e. a job which can run in any net).

CREATE-PROD-JOB is used to create those sorts of jobs in the JMDLIB which are marked JOB-TYPE=STD in the net descriptions.

Example of netwide production preparation

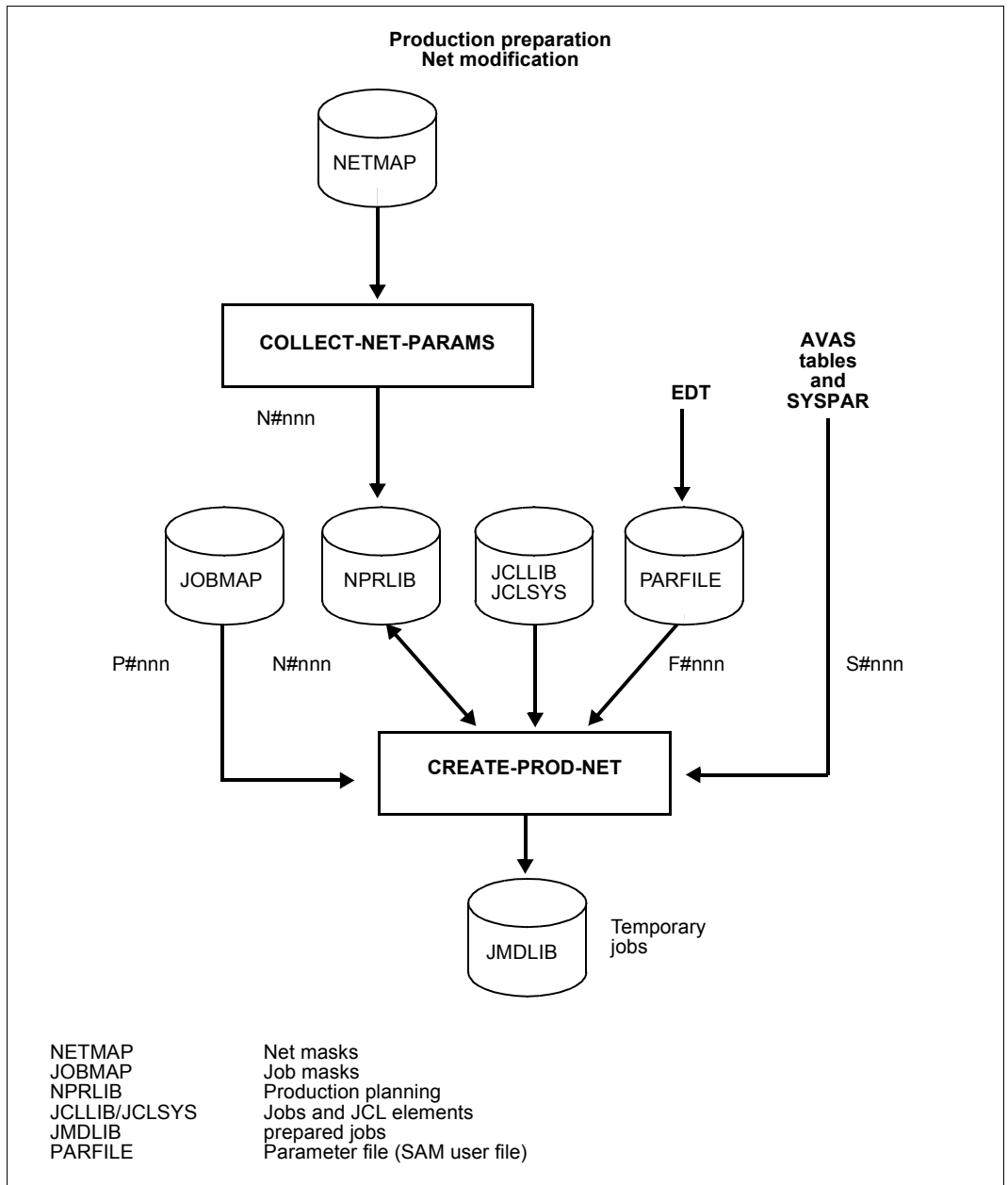


Figure 8: Production preparation: net modification

Example of job-oriented production preparation

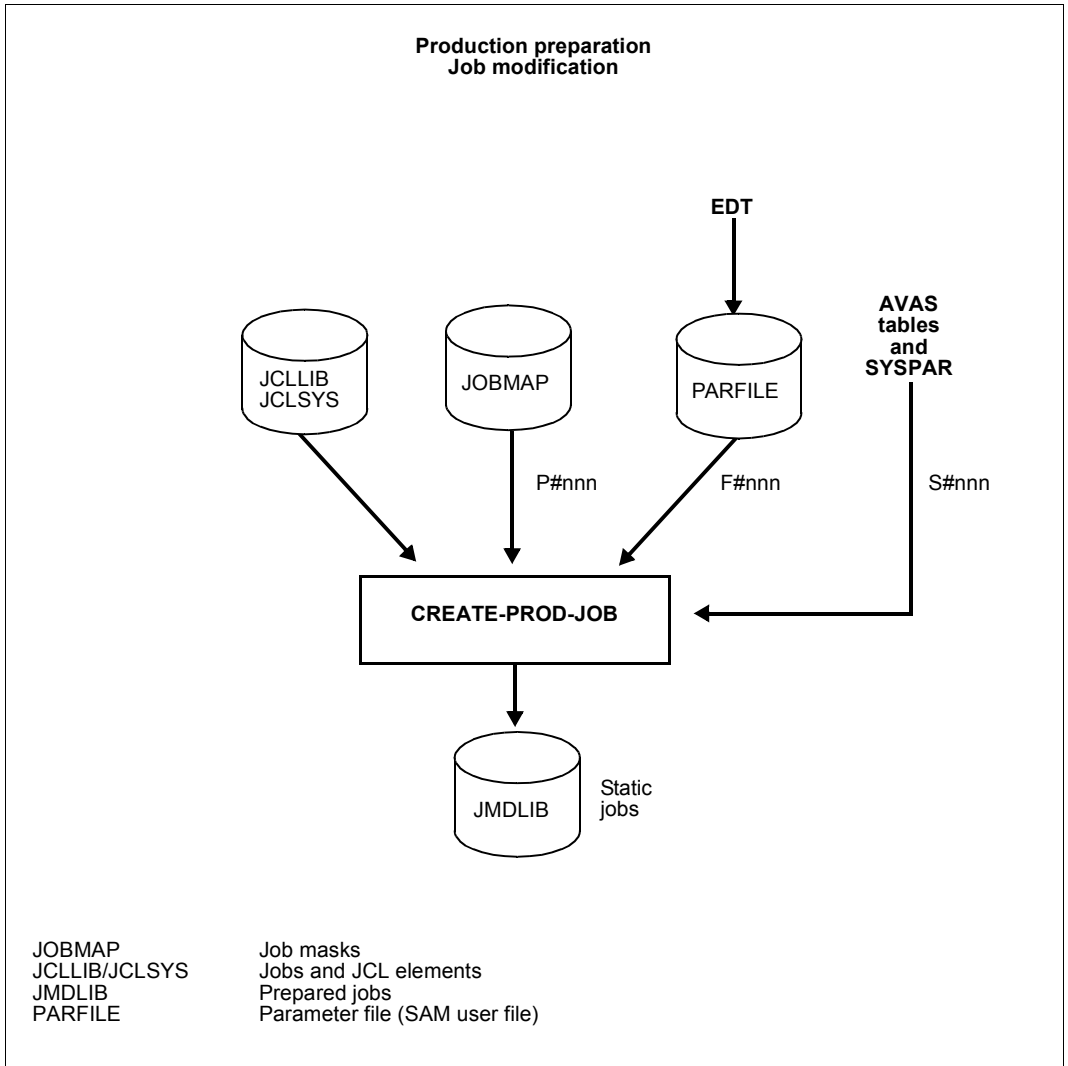


Figure 9: Production preparation: job modification

4.2.2 Time of modification

AVAS cannot perform modification until the net has been added to the production plan at the “plan DP production” stage and included in the library of planned nets (NPRLIB), thereby defining which jobs in the net are to be brought to execution.

During modification, the first thing that happens is that the net parameters to be entered via the net masks are collected (COL-NET-PARAMS statement). Once all the net parameters are available, the production jobs can be created using the CREATE-PROD-NET statement, in which case the job masks called in the jobs are presented and filled in by the user. At this processing stage, all AVAS parameters in the production jobs are replaced by or are assigned the current values (S procedures).

Assuming that the job masks apply only to the job in which they are called, the completeness check is performed separately for

- the parameters from the USER-PARAM-FILE,
- the parameters from the net masks (COL-NET-PARAMS) and
- the parameters from the job masks (#AVM# format).

Here it is assumed that the job masks are valid only for the job in which they are called.

If all jobs marked accordingly in the net description were created via CREATE-PROD-NET, the CREATED status is set for the net.

Before being released for production, the net must be completely supplied with current values for the run parameters.

If the net description specifies that a job is stored in a user file (JOB-TYPE=EXT), the user must make sure that this file is executable by the run control system at the time of the ENTER call.

4.2.3 Modifying hypernets and subnets

The subnets are automatically edited when hypernets are modified. The status of the subnet is shown in the structure element with FU=S and TYPE=NET in the hypernet.

Example

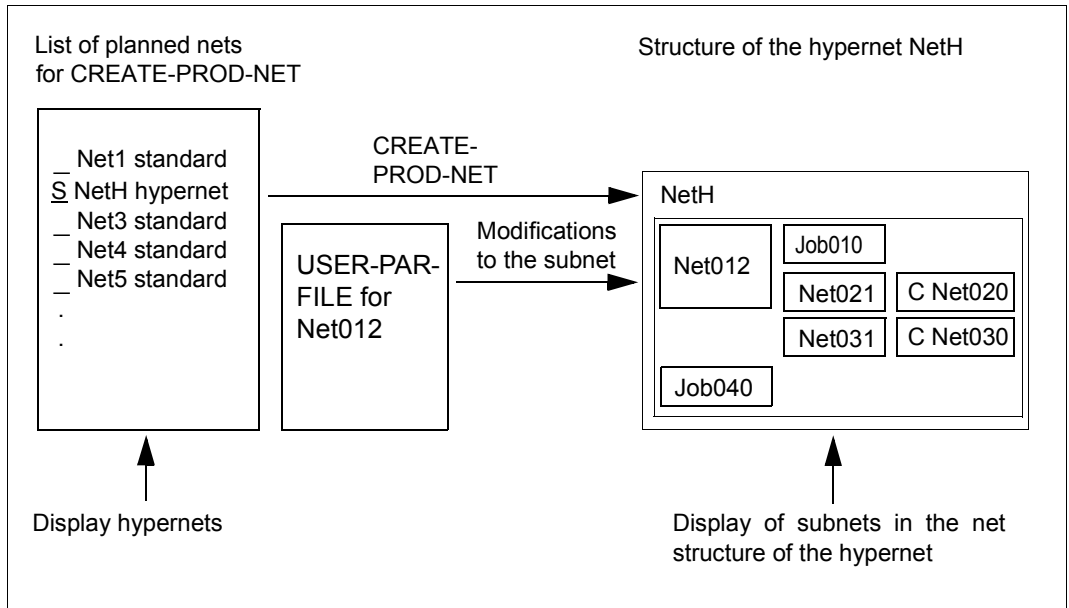


Figure 10: Example of editing the subnets via the hypernet with the CREATE-PROD-NET statement

The editing of the subnets via the hypernet has the following effects on the subnets and the jobs within the subnets:

- **DELETE-PROD-NET:** The modified jobs of the subnets are deleted with the jobs of the hypernet.
- **MODIFY-PROD-NET:** Modified jobs within subnets are deleted via the hypernet. If a subnet is marked with S in the structure of the hypernet, individual modified jobs of the subnet can be deleted. If a subnet is marked with Y in the structure of the hypernet, all modified jobs of the subnet will be deleted.

If subnets are modified directly with AVAS statements, any change to the status is not passed to the hypernet.

A planned subnet is uniquely assigned in a planned hypernet and cannot therefore be used in a different hypernet.

4.3 Release for production

Net processing starts as soon as the nets are released either automatically or explicitly by production preparation.

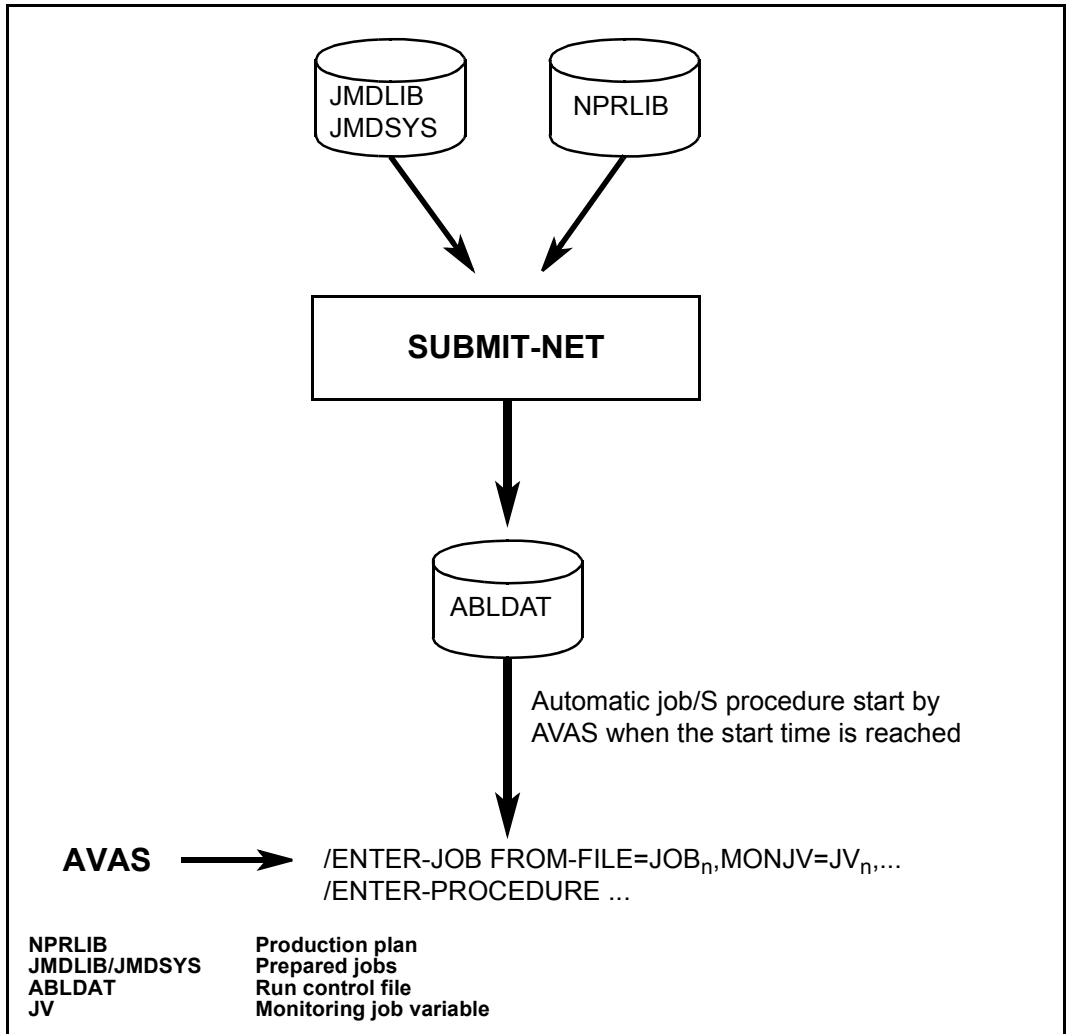


Figure 11: Release for production

4.3.1 Release of nets for processing

The following are prerequisites for the workstep “release of nets”:

- Processing of the net was planned in the “DP production planning” workstep (the net has been included in the production plan).
- All jobs which run in the net which use variable runtime parameters have been supplied with current values in the workstep “modify runtime parameters”.
- All jobs which run in the net which do not use variable parameters are contained in the library of the production job.
- The specified net has not yet been released.

The SUBMIT-NET statement releases a net for production. It covers the following activities:

- The structure description of the net is transferred to the job file.
- The JCLs of all jobs addressed within the net are transferred to the job file.
- The net is assigned to the specified run control system.
- Release of the net is noted in the production plan.
- Release of the net and the assigned jobs are logged in the journal file.

The result of the workstep “release of nets” is that the net is ready to be started, it is available in the job file and has been given the status SUBMITTED in the production plan. If a net is to be processed more than once after it has been planned, the second production release and all subsequent production releases must be carried out with the statement REPEAT-NET.

This statement covers the same activities as the SUBMIT-NET statement. In addition, a copy of the net is generated in the production plan. This is given the net status REPEATED.

4.3.2 Connecting AVAS and MAREN

If the interfacing module to the MAREN volume management system is used, it is possible to automatically ensure during release that

- all volumes required by planned jobs are available and
- that the current volume identifiers can be entered in jobs.

In addition, volume readying lists can be generated for the archive and the operator.

The following statements are available for release for production. These statements are described in the “AVAS Statements” manual [1].

REPEAT-NET Repeated release of a planned net

SUBMIT-NET Release planned nets

Subnets are automatically released when the associated hypernet is released. A hypernet can only be released when the subnets planned with the hypernet have also been modified. The subnets must be in the status CREATED or NOTTOCREATE.

4.4 Production control and execution

The AVAS run control system carries out production (control). It starts and monitors the nets and jobs, checks the conditions and logs the runs.

Statements for controlling and editing released nets

AVAS offers comprehensive display functions and intervention possibilities for monitoring and controlling the released production. The following statements are available which are described in the “AVAS Statements” manual [1].

CANCEL-NET	Interrupt or abort a net due to an error.
HOLD-NET	Suspend nets which are currently being processed.
MODIFY-COND-DESCRIPTION	Modify a condition description.
MODIFY-SUBMIT-JOB	Modify a job of a released net.
MODIFY-SUBMIT-NET	Modify a released net.
NET-CONTROL	Show and edit a released net.
RESTART-NET	Restart a net following an error.
RESUME-NET	Cancel the HOLD state.
SHOW-COND-DESCRIPTION	Display a condition description.
SHOW-NET-STATUS	Display the status of released nets or nets currently being processed.
START-NET	Start processing of a net regardless of the start conditions.

4.4.1 Run control for standard nets

Running nets

After release, the individual jobs are scheduled automatically.

The run control system starts the nets at the specified times, queuing the jobs in the order defined in the structure description and taking into account all declared dependencies on other nets, jobs, job variables, conditional values, and resources. In the process, AVAS monitors the normal and abnormal termination of each individual job.

Nets are processed via the run control and monitoring routine required for starting or restarting and for the termination of a net.

The start takes place in accordance with the start conditions. The AVAS run control system outputs the job/S procedures to a temporary ENTER file and then issues an /ENTER-JOB or /ENTER-PROCEDURE call. Prior to the call, the CC exit AVEX0401 is called (access to the JCL of a BS2000 job or of an S procedure). In a CC routine linked to this exit, both the parameters of the ENTER call and the ENTER file itself can be modified. Server jobs are also started with /ENTER-PROCEDURE via a special BS2000 agent (see [page 125](#)).

FT requests are executed using the /TRANSFER-FILE command and, like all jobs, are monitored using a MONJV. CC exit AVEX0401 is not called; no JCL exists.

Termination of a job/S procedure is recognized by the fact that the task job variables created by AVAS are monitored via event control. Here, besides the termination attributes set by the operating system, the additional information defined by the user in the task job variable is also checked. This additional information is passed to the task job variable via the AVAS statement #AVJ#, by a /SET-JV-LINK command or a SETJV macro with the corresponding format.

For each running job/S procedure, the termination information is kept in the run control file (ABLDAT) and in the journal (JRNDAT). Error-free termination of the jobs is a prerequisite for the further processing of the net. Jobs ending abnormally cause the net to stop with an error status.

A maximum waiting time can be defined to cater for situations in which nets or jobs do not start or conditions cannot be resolved in good time. Once this period has been exceeded, AVAS initiates the actions defined by the user (e.g. abort).

The MODIFY-SUBMIT-NET and MODIFY-SUBMIT-JOB statements make it possible to modify released nets and jobs/S procedures.

MODIFY-SUBMIT-NET permits changes to a net that has already been released. The net must have the net status WAITING, OPWAIT, HOLD, ERROR or NETWAIT (subnet).

Nets in the RUNNING status can only be edited if a structure element has already terminated with errors (status: CALLED FOR ERROR). In these nets, it is only possible to modify the structure elements with the status ERROR.

Modifications to jobs only take effect if they are performed before a structure element is started or restarted.

Modifications to the condition descriptions only take effect if they are performed before the conditions are fulfilled (status: OCCURRED) or before the restart (with the default value RESTART-WAIT-CONDITION=YES).

Modifications to the structure elements for editing condition descriptions only take effect if they are performed before the structure element is executed.

In this event, it is only possible to modify certain parameters:

- Net parameters,
- Parameters of structure elements for executing jobs,
- Parameters of structure elements for executing FT requests or
- Parameters of the condition descriptions

The JCL of a job in a net which has already been released can be modified with MODIFY-SUBMIT-JOB. The net must have the net status WAITING, HOLD, ERROR or NETWAIT (subnet).

Nets in the RUNNING status can only be edited if a job has already terminated with errors (status: CALLED FOR ERROR). In these nets, it is only possible to modify the JCL of the jobs with the status ERROR.

Before a modification is to be made, the HOLD-NET statement must be used to remove the net from the control of the run control system. After the modification has been made, it is made subject to monitoring again with the RESUME-NET statement.

The JCL of the jobs can be edited with EDT.

With S procedures, it is also possible to create or edit any procedure job parameters which may exist.

Processing of a net can be interrupted or aborted with the CANCEL-NET statement. The severity of the abortion can be determined using the CANCEL-TYPE parameter. After the statement has been executed successfully the net is in ABENDED or ERROR status.

CANCEL-NET can also be used to abort jobs which have already been started at operating system level. This function is controlled via the additional operand KILL-JOBS= YES/NO.

If a net is selected in the net overview, the operand applies equally for all jobs of the net, i.e. for BS2000 jobs and for FT requests. For an individual net it is also possible to abort an individual job in the net structure display (FU=J/P/F).

Restarting nets after error situations

If a net aborts due to an error situation in one of its elements (jobs, conditions), it is possible to restart the net. The restart variants defined for this purpose in the net description determine the point at which processing resumes. Therefore, in the event of a restart, individual jobs may be skipped or additional jobs may be executed.

The parallel structure defined in the net is not affected by the restart, i.e. even if the net is still running, a restart can be activated for independent net branches.

A restart can be initiated either automatically by AVAS or at the request of the user after checking the error situation.

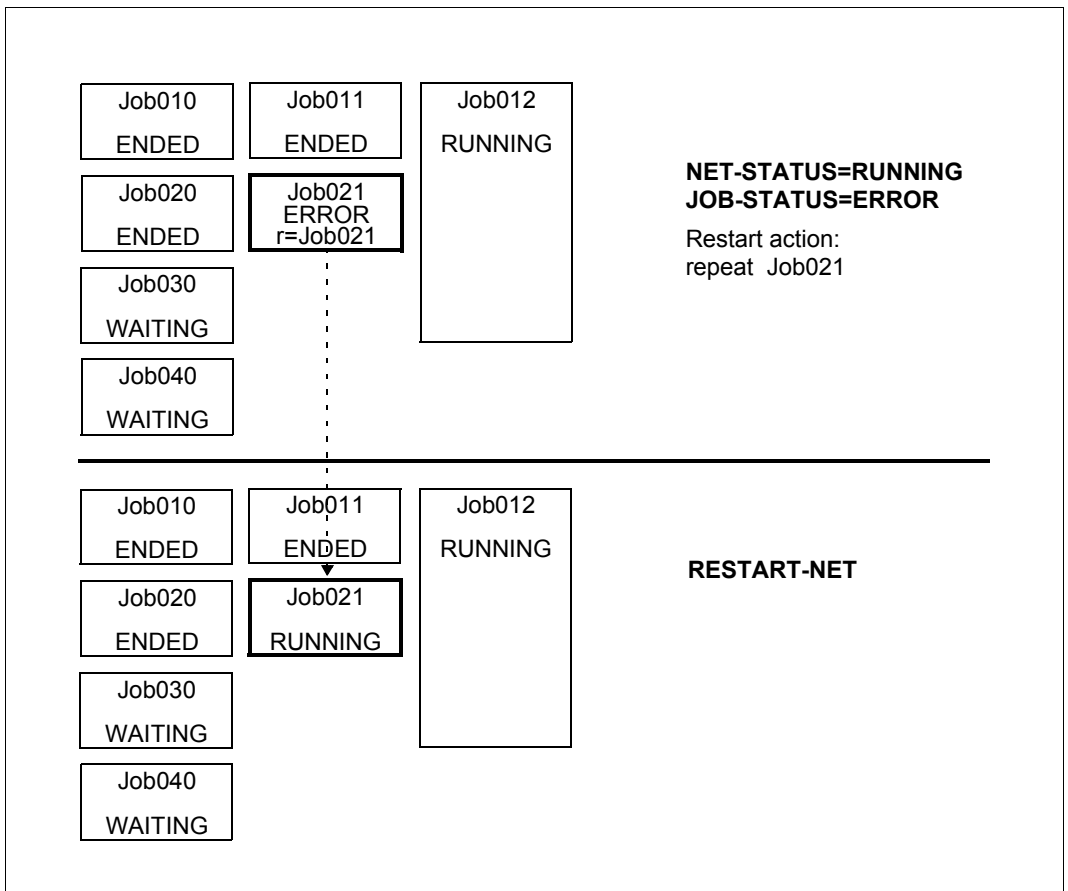


Figure 12: Example of a restart in an active net

In the example, JOB021 encounters and error. This job can be restarted immediately, i.e. while other jobs (JOB012) are running in a parallel net branch.

Abnormal terminations do not necessarily result in a negative status. You can define which status (or status list) leads to compliance with or violation of a condition. If there are no user-defined values, AVAS uses default values. Compliance with or violation of a condition can thus be programmed in advance.

4.4.2 Run control for hypernets

Hypernet are started and monitored in the same way as standard nets.

Subnets are started when the corresponding structure element of the hypernet with FU=S and TYPE=NET is processed.

The following rules apply to the processing of hypernets and subnets:

- If a net is started via a structure element with FU=S, the run control system sets up a monitor job variable (AVS-SUBNET-JVA). This job variable is used to implement the event-driven processing of the subnet (starting and monitoring).

The run control system of the subnet sets the status information RUNNING, CONDWAIT, HOLD, ENDED, ERROR or ABENDED in this job variable. The status of the hypernet is derived from this.

- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, the start of the subnet is initiated if the subnet is in the status NETWAIT:
 - The assigned AVAS-SUBNET job variable is set up (with the contents RUNNING/\$S)
 - The status NETWAIT is changed to the status WAITING in the subnet
 - The status RUNNING/\$S is set in the structure element

The subnet is started on the next cycle of the run control system assigned to the subnet if EARLIEST-START has been reached for the subnet.

Otherwise, the subnet remains in the status WAITING and the assigned structure element remains in the status RUNNING/\$S until EARLIEST-START has been reached. The subnet can then only be edited via the hypernet.

- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, this has the following effects if the subnet is in the status HOLD/NETWAIT:
 - The assigned AVAS-SUBNET job variable is set up (with the contents HOLD/\$S)
 - The status HOLD/NETWAIT is changed to the status HOLD/WAITING in the subnet
 - The status RUNNING is set in the structure element

The subnet is started on the next cycle of the run control system assigned to the subnet after #RESUME-NET has been executed via NET-CONTROL for the subnet. The subnet can then only be edited via the hypernet.

- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, the status IGNORED is set for the structure element with FU=S and TYPE=NET if LATEST-START has been exceeded for the structure element and if DELAY-SOLUTION=IGNORE was specified.

The subnet is not started and must be set to the status ABENDED with CANCEL-NET or must be executed independently of the hypernet.

- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, the status ENDED is set for the structure element with FU=S and TYPE=NET if the subnet is in the status ENDED.
- If a structure element with FU=S and TYPE=NET can be started because all dependencies have been fulfilled, the status ERROR is set for the structure element with FU=S and TYPE=NET if
 - the subnet is not in the ABLDAT
 - the subnet cannot be read, for instance because it is locked (RESULT=LOCKED)
 - The subnet is not in the status NETWAIT or ENDED
 - NET-TYPE>4 is not set for the subnet.

If the subnet is in the status ERROR, this has the following consequences:

- No AVAS-SUBNET job variable is set for the subnet.
- The subnet cannot be started and edited via the hypernet.
- The subnet must be put into the status ENDED or ABENDED by the user.

If after a manually initiated start (for NET-TYPE < 4), the subnet reaches the status ENDED, the status ENDED can be passed to the hypernet by executing the statement RESTART-NET with the parameters RESTART-INDEX=ERROR-INDEX for the hypernet.

- The status shown for the structure element with FU=S and TYPE=NET for starting a subnet only corresponds to the status of the subnet itself if the subnet was started via the hypernet (status RUNNING in the structure element) or if the status ENDED is displayed for the structure element..

In all other cases, the status of the subnet is undefined, as the subnet can be edited and executed independently of the hypernet.

If an attempt is made to put a hypernet into the status ABENDED with CANCEL-NET and CANCEL-TYPE=HARD, a warning is issued and processing can be canceled with RETURN.

This message is not issued for standard nets. If they are marked with Y, the nets are put in the ABENDED status without anything being displayed or any message being issued.

LATEST-START and DELAY-SOLUTION parameters

The following needs to be taken into account during planning with regard to the LATEST-START and DELAY-SOLUTION parameters:

A distinction must be made between the two cases

- LATEST-START exceeded for a structure element with FU=S and TYPE=NET and
- LATEST-START exceeded for the net parameters of a subnet.

The run control system of the hypernet checks whether LATEST-START has been exceeded for a structure element FU=S/NET. The run control system of the hypernet does not check the parameters of the subnet.

The run control system of the subnet checks whether LATEST-START has been exceeded for the net parameters of a subnet.

On activating the structure element with FU=S/TYPE=NET, the run control system of the hypernet changes the status of the subnet from NETWAIT to WAITING and sets the status RUNNING/\$S in the AVAS-SUBNET job variable and in the structure element with FU=S and TYPE=NET.

The run control system of the hypernet detects whether LATEST-START has been exceeded for a structure element with FU=S and TYPE=NET and carries out appropriate processing.

Depending on DELAY-SOLUTION in the structure element with FU=S and TYPE=NET, the status parameters in the structure element (SE), hypernet and subnet are set as follows:

DELAY-SOLUTION	STATUS of SE with FU=S / TYPE=NET	NET-STATUS in hypernet	NET-STATUS in subnet
START	RUNNING/\$S	RUNNING	WAITING ⁽¹⁾
IGNORE	IGNORED	RUNNING	IGNORED ⁽²⁾
CANCEL	ERROR	ERROR	ABENDED ⁽²⁾

¹ The run control system of the subnet starts the net or sets the status in accordance with DELAY-SOLUTION in the net parameters of the subnet.

² If DELAY-SOLUTION=IGNORE or DELAY-SOLUTION=CANCEL, the subnet is not processed by the run control system of the subnet. The subnet remains in the status NETWAIT must be put into the status ENDED or ABENDED by the user.

If the subnet is to be executed independently of the hypernet, the user must use MODIFY-SUBMIT-NET to change NET-TYPE>4 to NET-TYPE<4. This changes the status NETWAIT to WAITING and the (sub)net can be processed independently of the hypernet.

If the subnet is not to be executed, the user must use CANCEL-NET with the parameter CANCEL-TYPE=HARD to put it into the status ABENDED.

The run control system of the subnet checks LATEST-START and DELAY-SOLUTION prior to starting. Depending on DELAY-SOLUTION, the status parameters in the subnet, structure element (SE) and hypernet are set as follows:

DELAY-SOLUTION	NET-STATUS in subnet	STATUS of SE with FU=S / TYPE=NET	NET-STATUS in hypernet
WAIT	WAITING	RUNNING/\$S	RUNNING
START	RUNNING	RUNNING	RUNNING
IGNORE	IGNORED	IGNORED	RUNNING
CANCEL	ABENDED	ERROR	ERROR

Note

If the planning of the subnet is not done on the basis of a separate symdat, but using the parameters of the structure element with FU=S and TYPE=NET, DELAY-SOLUTION =IGNORE or DELAY-SOLUTION=CANCEL are passed to the subnet and then DELAY-SOLUTION=START is set for the structure element. This ensures that the subnet is started and or that the status is set in accordance with DELAY-SOLUTION.

Dialog functions for run control and monitoring of the hypernet

The statement NET-CONTROL is used for interactive control and monitoring of hypernets. Subnets are not shown on the net overview with NET-CONTROL.

To edit a subnet, the structure element is marked with S on the structure mask of NET-CONTROL (AVI023) and the operation is initiated with CMD. After returning to NET-CONTROL, the assigned status is set in the hypernet if necessary. The operation #NET-CONTROL (#33) is used to branch from the hypernet structure to the subnet structure via the mark S.

All other statements display all types of nets (hypernets, standard nets, subnets), but do not allow the editing of structure elements with FU=S.

Restrictions when editing released subnets

The following must be observed when modifying a released net with MODIFY-SUBMIT-NET:

- It is only possible to change NET-TYPE>4 to NET-TYPE<4 (remove the subnet property) if the subnet is in the status NETWAIT. If the net has been started, it remains under the control of the run control system of the hypernet until the status ENDED or ABENDED has been reached. If the subnet was started via the hypernet, the run control system of the subnet can no longer be modified.
- It is only possible to change NET-TYPE<4 to NET-TYPE>4 (assignment of the subnet property) if the name of a hypernet is stored in the net (this is done when a structure element with FU=S and TYPE=NET is started) and the net is in the status WAITING. If the net was not started via the hypernet, it can no longer be monitored via the hypernet. The structure element with FU=S and TYPE=NET remains in the status WAITING until all dependencies have been resolved and then changes to the status ERROR.
- The name of the run control system cannot be changed for hypernets.

5 Production monitoring

Net processing can be monitored at all stages of the production run using the SHOW-NET-STATUS statement. Thus, the progress of an individual net or of the entire production run can be monitored. Similarly, monitoring can also encompass all nets with a certain processing status. The documentation associated with nets and jobs can also be displayed.

The NET-CONTROL statement allows you to use operations to branch to any statement that can be used for editing or controlling a released net. All possible operations are permitted on the overview of the nets and for the display of the net structure.

Benefit to the customer: Simplifies the user interface for the run control system. This makes it much easier for AVAS users to perform production monitoring and the diagnosis of errored job runs.

Production monitoring statements

The following statements are available for production monitoring. These statements are described in the “AVAS Statements” manual [1].

NET-CONTROL	Display and edit released nets
SHOW-JOB-LOG	Display job logs
SHOW-JOURNAL	Display journal records
SHOW-NET-STATUS	Display status of released nets or nets currently being executed
START-MONITOR	Call the AVAS status monitors

5.1 Monitor

Driven by events, the start and end of a net are displayed together with any faults that occur in the process, i.e. when net elements are unexpectedly placed in awaiting state due to error situations or non-compliance with conditions.

The monitor is called with the START-MONITOR statement. Filter functions can be used to determine the scope of the data (nets) displayed. This can involve different events such as obtaining the

- net statuses ABENDED, CONDWAIT, ENDED, ERROR and HOLD.
- job/condition statuses ERROR, NO-OCCURE, OCCURED, RESTARTED and SKIPPED.

5.2 Journal file

For control purposes, the monitoring of all DP production runs handled by the AVAS system is based on the production log, also known as the journal file. It contains all user activities and all actions of the AVAS systems for the processed nets.

In the AVAS journal, all actions are logged from the production planning stage onwards. Any operations which cannot be assigned uniquely to a net (e.g. changes made to static jobs) are not logged.

Actions on FT requests are treated like the corresponding actions on jobs.

The following events and operations are logged during the individual work stages.

- Plan DP production
 - Net control record with resolved real start time
 - Control records of the jobs/S procedures whose execution is contingent upon the processing cycle and upon symbolic start dates
- Edit and delete temporary jobs/S procedures in the JMDLIB
- Modify the parameters of temporary jobs/S procedures
 - complete name of the job
 - name of a user mask or a parameter file of the user
 - inclusion of a JCL element in a job/S procedure
 - replacement of the parameters with current values
 - error message

- Release nets for processing
 - release for production
 - no release for production due to error
 - cause of error
- Modify released nets
 - modify net control records
 - modify job control records
 - modify condition control records
 - modify the BS2000 statements in the jobs/S procedures
 - modify a released net (MODIFY-SUBMIT-NET)
 - modify a job/S procedure of a released net (MODIFY-SUBMIT-JOB)
- Run control system
 - wait for real start time of a net to be reached (WAITING)
 - latest start time of a net reached (DELAY-SOLUTION)
 - start net (START-NET)
 - start job/S procedure (START-JOB)
 - end job/S procedure (END-JOB)
 - halt net due to unresolved dependency on job level (CONDITION-WAIT)
 - resolve dependency of a net (CONDITION-OCCURRED)
 - halt net via a command (HOLD-NET)
 - resume halted net (RESUME-NET)
 - error message job/S procedure (ERROR-JOB)
 - interrupt net due to error (ERROR-NET)
 - resume net following error (RESTART-NET)
 - execute RESTART statements (UPDATED)
 - execute restart jobs (RESTART-JOB)
 - end net (END-NET)
 - abort net processing (CANCEL-NET)

As a rule, the net/job control records involved are logged, as is the corresponding error text in the case of error messages.

The SHOW-JOURNAL statement enables the user to display the entries in the journal file. While this statement is being processed, the job runtime logs are also provided.

5.3 SYSOUT file

During the active phase of an AVAS job its SYSOUT file can be accessed: the OUTSYS (#79) operation of the NET-CONTROL command opens the file and transfers it to EDT.

Access to the SYSOUT file is implemented by the AVAS task SOUT which is started with privileges: it identifies the file via BS2000 system interfaces, opens it, copies it temporarily and transfers it to the AVAS dialog task.

SOUT task

The SOUT task is a shareable DCAM application. It consists of a primary task and a certain number of secondary tasks. The primary task provides the resources of the central DCAM task and controls the application.

The secondary tasks accept the requests from the AVAS dialog tasks, access the SYSOUT files and transfer them.

The SOUT task is started on every BS2000 host on which AVAS jobs run using an ENTER call.

5.4 History file

In addition to the AVAS journal, AVAS also maintains a file containing historical data in compressed form. The following records are stored:

- a record for each job or FT request executed under a net
- a record for each executed net
- a compressed record containing average values for each planned variant of all nets
- a compressed record containing average values for every job

The history data is derived from the AVAS journal data. The structure and the updating of the file is performed when the journal file is reorganized (SJOUR function). The average values are determined when a run is entered and are updated in the records containing the compressed values. A new variant - #AVA#\$H (entering history data in the journal file) - of the AVAS statement #AVA# allows the user to transfer data such as the CPU load, I/O rate, host name, or other user information from the job to the HISTORY file for storage. Display of the history data is linked with the NET-CONTROL statement.

Benefit to the customer: Easy way of estimating the probable runtime for job production (forecasting). Saves the time involved in creating the users' own evaluations.

5.5 Runtime logs

The runtime logs of BS2000 jobs started via the AVAS run control system can be collected via AVAS and stored in a central library.

Two programs, which have to be integrated in the job, are used for transferring the job runtime logs:

- the SIGNAL program notifies AVAS that a log is to be transferred
- the TRANSFER program transfers the log from the user area to the AVAS area

The DCAM application CENTRAL, which accepts the log and stores it in the log library (AVAS pool), has been added to the AVAS system.

The interactive and batch statements ADD-JOB-LOG are available for subsequent storage of signaled logs.

A maximum of 99 logs can be signaled and stored for each job run.

The logs can be displayed using the program EDT by means of the SHOW-JOB-LOG, SHOW-NET-STATUS and SHOW-JOURNAL statements.

As part of reorganization the logs can be saved and deleted.

Individual logs can also be deleted using the interactive or batch statement DELETE-JOB-LOG.

Collecting the logs

A DCAM application is available for storing logs centrally at the time of the job run. The application consists of a number of DCAM programs which communicate with one another. The SIGNAL and TRANSFER programs notify the central task CENTRAL that the logs are available for transfer and then transfer them for central storage.

CENTRAL task

CENTRAL is a shareable DCAM application.

CENTRAL consists of a primary task and a certain number of secondary tasks. The primary task provides the resources for the central DCAM task and controls the application. The secondary tasks take on the requirements of the DCAM programs and store the logs at a central location.

In addition to the central access tasks (ZDs) the CENTRAL task is started by means of an ENTER call on the same mainframe and under the same user ID. CENTRAL logs on to the ZDs as a special task (in the same way as AVAK).

SIGNAL program

SIGNAL is a non-shareable DCAM program.

Immediately after the /SET-LOGON-PARAMETERS statement and the /ASSIGN-SYSOUT statement, SIGNAL is loaded as the first program in an AVAS job. It ascertains the AVAS job ID and the name of the file with the SYSOUT information and sends this information together with other control parameters to CENTRAL. SIGNAL indicates any problems in a program job variable.

If further logs (e.g. SYSLST) are to be stored centrally, the SIGNAL program must be called for each log.

TRANSFER program

TRANSFER is a non-shareable DCAM program.

Either it is loaded in the AVAS job immediately before the LOGOFF statement as the last program or it runs, after being activated by AVAS, as an independent job and single program.

TRANSFER ascertains the AVAS job ID and sends this information together with any specified parameters to CENTRAL. If the information is correct, CENTRAL requests the TRANSFER program to transfer the log data. Any problems which occur in the TRANSFER program are reported in a program job variable.

5.6 Reports

Reports are used for the retrospective monitoring of production.

For example, in the case of daily inspection, the report function generates the following standard reports:

- PLANNED-NET-MODIFICATION report containing all unplanned modifications to nets and jobs which were carried out after planning.
- OUT-OF-PLAN report containing all planning deviations that occurred during net handling, e.g. delays and error situations.

A parameterization option permits, for each report, the definition of the work files, the user groups to be included, the sort sequence, and the net status values to be taken into account. The display of negligible delays relative to the net start time can be suppressed by the specification of threshold values. Nets in the current production plan can be selected using NET-NAME and PERIOD-NAME.

OUT-OF-PLAN report

The OUT-OF-PLAN report exhibits in compressed form all the nets that deviate from the plan specifications.

The report covers the following groups:

- Abnormally terminated nets (with or without delay)
- Delayed nets at the preparation stage
- Delayed nets at the execution stage
- Nets not executed (because their latest start time was exceeded)

A check is made for each net, on the basis of the journal, whether ABENDED or ERROR was reached as the last AVAS status, i.e. whether an abnormal termination occurred. If so, the net is included in the report. The following basic items are listed: the net name, the actual and planned start and end times, the AVAS system ID, the user ID, and the job names (with contents of task job variable) responsible for the error. By default, the system termination identifier (\$T, \$A, etc.), the task sequence number (TSN) and the AVAS restart variant (RV=0, 1, 2 or 3) are displayed. The optional user section may contain additional information such as the work step (STEP), program name or error code.

Delayed nets at the preparation stage are nets for which the earliest or latest start time is less than the current start time and which have not yet been released, i.e. which have the status TOCREATE, PARTIALLY-CREATED, CREATED or NOTTOCREATE.

Delayed nets at the execution stage are nets whose actual start time has exceeded the earliest or latest start time. The special states “waiting for start”, “executing” or “normally terminated” are output.

Nets not executed are nets assigned the IGNORED status by AVAS via the DELAY-SOLUTION option IGNORE because their latest start time was exceeded.

Nets restarted after ERROR or terminated normally after a restart are not regarded as abnormal.

Delays are shown only if they are greater than the defined threshold value.

PLANNED-NET-MODIFICATION report

The PLANNED-NET-MODIFICATION report comprises all the nets that were modified after completion of production planning.

A net is included in the report if any of the statements MODIFY-PLAN-NET, EDIT-PROD-JOB, MODIFY-SUBMIT-NET or MODIFY-SUBMIT-JOB was issued or if parameters were changed or jobs excluded from processing via SUBMIT-NET.

For each change, a line with the following contents is shown:

- the type of change and the statement used to execute it,
- the time of change,
- the user who initiated the change, and possibly,
- the object concerned (NET-NAME, JOB-NAME, FT-NAME, CONDITION-NAME).

If SUBMIT-NET is used to exclude jobs or conditions, a separate line is generated for each object concerned.

6 Status of the networks during processing

This chapter contains overviews of the net statuses for AVAS statements, the log statuses for AVAS statements and the statuses of the condition descriptions for AVAS statements (grouped by condition type).

6.1 Overview of net statuses for the statements

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
CREATE- PLAN-NET	–	TOCREATE NOTOCREATE	–	–	The net is incorporated in the production plan.
MODIFY- PLAN-NET	TOCREATE NOTTOCREATE PARTIALLY CREATED	TOCREATE NOTTOCREATE PARTIALLY CREATED	–	–	The net status is not modified.
COLLECT- NET-PARAMS	TOCREATE	TOCREATE			The net status is not modified.
CREATE- PROD-NET	TOCREATE PARTIALLY	PARTIALLY	–	–	Not all tasks in the net are produced.
	TOCREATE PARTIALLY	CREATED	–	–	All tasks in the net are produced.
MODIFY- PROD-NET	PARTIALLY CREATED	PARTIALLY	–	–	If at least one task is deleted, even if it was the last one, the status is set to PARTIALLY.
DELETE- PROD-NET	PARTIALLY CREATED	TOCREATE	–	–	All net tasks in the JMDLIB that have been modified up to now are deleted.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
SUBMIT-NET	CREATED NOTTOCREATE	SUBMITTED	-	WAITING	The net is copied from the NPRLIB to the ABLDAT. Afterwards the net is in both the NPRLIB and the ABLDAT
				OPWAIT	The net waits for the START-NET statement. Afterwards the net is in both the NPRLIB and the ABLDAT.
				NETWAIT	The net waits to be started by the hypernet Afterwards the net is in both the NPRLIB and the ABLDAT.
REPEAT-NET	-	REPEATED	-	-	A copy of a net with the SUBMITTED status is created with a new scheduled start time (NEW-PLAN-START).
				WAITING	The net waits for the start time to be reached.
				OPWAIT	The net waits for the START-NET statement.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
Net start via the run control system	SUBMITTED REPEATED	SUBMITTED REPEATED	WAITING START	RUNNING	At least one task is started on the first index level.
				CONDWAIT	At least one task is started on the first index level.
				HOSTWAIT	The net waits for a host in the MSCF network. No tasks are currently executing.
				ERROR	Structure elements on the first index level cannot be processed without error.
			NETWAIT	WAITING	Structure element with FU=S/TYPE=NET is being activated. The start of the subsystem is being initiated.
LATEST- START time passed	SUBMITTED	SUBMITTED	WAITING	WAITING	WAIT in the DELAY-SOLUTION parameter in the NET-DESCRIPTION For nets which are waiting for this net, the condition is not satisfied.
				IGNORED	IGNORE in the DELAY-SOLUTION parameter in the NET-DESCRIPTION For nets which are waiting for this net, the condition is satisfied.
				RUNNING	START in the DELAY-SOLUTION parameter in the NET-DESCRIPTION
				ABENDED	CANCEL in the DELAY-SOLUTION parameter in the NET-DESCRIPTION

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
HOLD-NET	SUBMITTED	SUBMITTED	RUNNING CONDWAIT HOSTWAIT RESTARTED START NETWAIT	HOLD	The net is given HOLD status if all tasks with RUNNING status have terminated.
				HOSTWAIT	At least one job in the net is waiting for a host which is currently inactive.
			WAITING NETWAIT OPWAIT ERROR	HOLD	The net is not subject to control by the run control system.
START-NET	SUBMITTED	SUBMITTED	OPWAIT WAITING	START	The net is brought to execution without taking the start time into account (only up to the next restart point of the run control system).
RESUME-NET	SUBMITTED	SUBMITTED	HOLD	RUNNING	At least one task on the following index level was started.
				CONDWAIT	The system waits at the following index level for a condition to be satisfied.
				HOSTWAIT	The net waits for a host in the MSCF network. No tasks are currently executing.
				WAITING OPWAIT RESTARTED START ERROR NETWAIT	The net is not subject to control by the run control system.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
CANCEL-NET	SUBMITTED	SUBMITTED	RUNNING	ERROR	CANCEL-TYPE=SOFT Active nets are given ERROR status when running tasks have terminated.
			CONDWAIT	CONDWAIT	CANCEL-TYPE=SOFT Inactive nets retain their net status.
			NETWAIT WAITING HOLD ERROR ENDED SHIFTED OPWAIT RESTARTED START	NETWAIT WAITING HOLD ERROR ENDED SHIFTED OPWAIT RESTARTED START	CANCEL-TYPE=SOFT The nets with the specified status are not displayed and cannot be processed.
			RUNNING CONDWAIT HOSTWAIT NETWAIT WAITING HOLD ERROR OPWAIT RESTARTED START	ABENDED	CANCEL-TYPE=HARD Nets subject to control by the run control system are given ABENDED status.
			NETWAIT	ABENDED	CANCEL-TYPE=HARD Subnets which have not been started are given the ABENDED status

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
MODIFY- SUBMIT-NET	SUBMITTED	SUBMITTED	HOLD ERROR WAITING OPWAIT START	HOLD ERROR WAITING OPWAIT START	The nets are inactive and their status is retained.
			RUNNING/ ERROR	RUNNING/ ERROR	The nets are under the control of the run control system. Only structure elements with the status ERROR may be processed. The nets retain their status.
			NETWAIT	NETWAIT	Subnets which have not been started retain their status
			NETWAIT	WAITIING	Subnets have not been started. Changing NET-TYPE > 4 to < 4 removes them from the control of the hypernet.
MODIFY- SUBMIT-JOB	SUBMITTED	SUBMITTED	HOLD ERROR WAITING OPWAIT START	HOLD ERROR WAITING OPWAIT START	The nets are inactive and their status is retained.
			RUNNING/ ERROR	RUNNING/ ERROR	The nets are under the control of the run control system. Only tasks with the status ERROR may be processed. The nets retain their status.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
Net processing via the run control system	SUBMITTED	SUBMITTED	WAITING START	RUNNING	Released net was started
				ERROR	At least one structure element of the net was processed with errors.
				ENDED	Net terminated without errors.
				HOSTWAIT	At least one structure element of the net waits for activation of a host or server.
				CONDWAIT	At least one structure element of the net waits for a condition.
			NETWAIT	WAITING	Subnet was released for execution by the hypernet.
			RUNNING	ENDED	Net terminated without errors.
				ERROR	Errors occurred during processing of at least one structure element of the net.
				HOSTWAIT	At least one structure element of the net waits for activation of a host or server.
				CONDWAIT	At least one structure element of a net waits for a condition.
			HOSTWAIT	RUNNING	Net was continued
			CONDWAIT	RUNNING	Net was continued

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
RESTART-NET	SUBMITTED	SUBMITTED	ERROR	RESTARTED	Until the next restart point of the run control system.
				RUNNING	After startup by the run control system if a task on the restarted index level was started.
				CONDWAIT	After startup by the run control system if the system is waiting for a condition to be satisfied at the restarted index level.
				HOSTWAIT	If the remote system is not available.
			RUNNING/ ERROR	RUNNING/ RESTARTED	Until the next restart point of the run control system.
				RUNNING	After startup by the run control system if the tasks at the restarted index level were restarted.
				RUNNING/ ERROR	After startup by the run control system if not all the structure elements with a status of ERROR were processed.
				ERROR	The processing of structure elements at the restart index level has again produced an ERROR status.
Reorganizing the ABLDAT	SUBMITTED	SUBMITTED	ENDED ABENDED SHIFTED IGNORED	–	The net is delete from the ABLDAT.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
DELETE- PLAN-NET	SUBMITTED REPEATED	–	–	–	The net must have been deleted from the ABLDAT (reorganizing ABLDAT). The net is deleted from the NPRLIB; it is removed from the production plan.
	TOCREATE PARTIALLY CREATED NOTTOCREATE	–	–	–	The net has not yet been released via SUBMIT-NET. The net is deleted from the NPRLIB; it is removed from the production plan.

6.2 Overview of log statuses for the statements

Statement/ Function	LOGSYS		Comments
	Old status	New Status	
AVAK	–	CREATED	A log entry is created by the run control system.
SIGNAL	–	ASSIGNED	A log was signaled by SIGNAL during job execution.
TRANSFER	ASSIGNED	TRANSFERRED	The log was transferred during job execution.
	ASSIGNED	ERROR	Log could not be transferred during job execution. An error occurred.
ADD-JOB-LOG	CREATED	ADDED IGNORE	An initial log was added to this job run. No log data should be added to this log entry.
	ASSIGNED	ADDED IGNORE	A signaled log was added. Signaled log is no longer relevant.
	ERROR	ADDED IGNORE	A signaled log, which could not be transferred during job execution, was added. A signaled log is no longer relevant.
	ADDED	ADDED	A new or additional log was added.
DELETE-JOB-LOG	ADDED ASSIGNED CREATED ERROR IGNORE SAVED TRANSFERRED	–	The log or logs relating to a job run were deleted. No further processing of the log for this job run is possible.

6.3 Overview of the status of the condition descriptions for each statement

COND-TYPE=NET

Statement/ Function	ABLDAT		Comments
	Old status	New Status	
SUBMIT-NET	–	CREATED	A condition description was created.
AVAK	CREATED	ENDED IGNORED ABENDED	Net processing terminated correctly. Due to the specified time having been passed, the net was not started (DELAY-SOLUTION=IGNORE). The net was not terminated correctly or was not started due to the specified time having been passed (DELAY-SOLUTION=CANCEL).
CANCEL-NET	CREATED	ABENDED	The net was terminated with the dialog function CANCEL-NET CANCEL-TYPE=HARD.
DELETE-COND-DESCRIPTION	CREATED/ ENDED/ ABENDED/ IGNORED	–	The condition description was deleted.

COND-TYPE=JOB

Statement/ Function	ABLDAT		Comments
	Old status	New Status	
SUBMIT-NET	–	NO-PLAN NO-SUBMIT CREATED	The task was excluded during planning. The task was excluded from a SUBMIT-NET. A condition description was created.
MODIFY-SUBMIT-NET	CREATED	DELETED	The task was excluded by a MODIFY-SUBMIT-NET.
AVAK	CREATED	ENDED ERROR IGNORED ABENDED	The task terminated normally. The task terminated with errors (ERROR status). The task was ignored because the specified time had passed (DELAY-SOLUTION=IGNORE). The task terminated with errors (ABENDED status).
RESTART-NET	ERROR/ ENDED/ SKIPPED/ CREATED/ ERROR	CREATED SKIPPED	The task is running again as a restart. The condition description was reset by the RESTART-NET. The task has been skipped during the RESTART-NET.
CANCEL-NET	CREATED	ERROR ABENDED	The task was abandoned due to a CANCEL-NET CANCEL-TYPE=SOFT. The task was abandoned due to a CANCEL-NET CANCEL-TYPE=HARD.
DELETE-COND-DESCRIPTION	CREATED/ ENDED/ ABENDED/ SKIPPED/ IGNORED/ NO-PLAN/ NO-SUBMIT/ DELETED	–	The condition description was deleted.

7 Multiprocessor operation with AVAS

In addition to local BS2000 hosts, the AVAS production control functions also supports BS2000 hosts in an MSCF cluster, as well as remote BS2000 systems. FT requests can transfer files to remote hosts which are configured as partner systems in *openFT*.

7.1 BS2000 multiprocessor operation with AVAS

The multiprocessor system HIPLEX MSCF (see the “HIPLEX MSCF” manual [3]) allows different mainframes to be interconnected to form a computer network.

In AVAS multisystem operation, one system is the AVAS master where the AVAS system including the run control system are located. AVAS can distribute entire nets or individual job from a net over an optional number of systems within a HIPLEX MSCF network and monitor the processing of the distributed jobs. A shared disk ensures that the job (ENTER) files are accessible to all systems.

Following a system failure the job load and, if the AVAS master fails, AVAS itself are automatically relocated to an active system.

The HIPLEX MSCF network also allows server jobs to be distributed. This is necessary when an AVAS slave processor rather than the AVAS master is connected to the server system.

The server representative in BS2000 must then be started on the slave processor with the connection to the server system.

Partner systems for file transfer using FT requests must be entered in *openFT*. The accessibility and status of the computers are not known to the run control system.

Configuration

In multiprocessor operation with AVAS, one of the processors in the HIPLEX MSCF network is used as the AVAS master processor. This handles the AVAS dialog and the run control system; jobs can also be processed using the master processor. The other processors in the HIPLEX MSCF network only perform processing of jobs (AVAS slave processor).

During processing of the nets, the AVAS run control system outputs the individual jobs or S procedures of the net to an ENTER file and issues an ENTER-JOB or ENTER-PROCEDURE call for this file. The call uses the parameters stored in the net/job description, i.e. in particular the catalog ID for addressing the target processor (NET-CAT, JOB-CAT). At the time of ENTER, AVAS determines from the MRSCAT the host names associated with the catalog ID to be specified, and thus passes the ENTER call.

If *ANY is entered as the catalog ID, AVAS determines the system in the XCS network with the lowest batch load using the JMS distribution components and starts the job on this system.

*ANY can only be specified if the system on which AVAS is started belongs to an XCS network.

If this is not the case, the job is assigned the status ERROR.

For details of the criteria for selecting the system with the lowest batch load, see the description of ENTER-JOB ... HOST=*ANY in the Appendix of the "HIPLEX MSCF" manual [3].

In order to permit access from the target processor to the ENTER procedure, AVAS stores this procedure either on a shared private disk (SPD) or on a shared pubset (also shared public volume set SPVS). In the master processor the standard catalog ID (STD-CATID) is used to output the ENTER file.

The job is then started on the processor on which the specified catalog is located at the time the ENTER call is made by the run control system. The monitoring job variables are created by AVAS on the relevant home pubset.

The AVAS data (ABLDAT, JRNDAT, libraries) must be directly accessible to the AVAS master processor. If the data is held on the shared pubset or SPD, AVAS can then be loaded on a host processor in the event of protracted failure of the AVAS master processor. If the data is held on a pubset (other than the home pubset), this pubset must be imported onto the other processor in the event of protracted failure of the AVAS master processor.

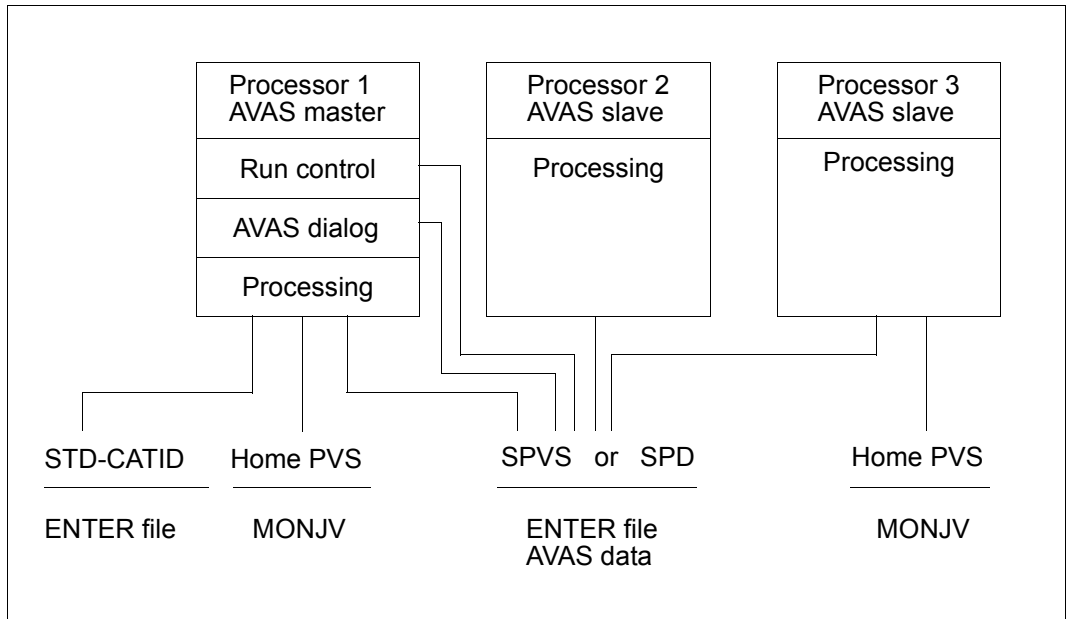
Sample configuration

Figure 13: AVAS in multiprocessor operation

All the systems in the MSCF network (BS2000 command /SHOW-MASTER-CATALOG-ENTRY, entries with HOST names and the attribute REMOTE-HOME), to which AVAS tasks are to be distributed, must be available. For monitoring purposes, the MSCF job variable with the name \$TSOS.SYS.MSCF.CONTROL-STATE.<hostname> is read and evaluated as follows:

- The MSCF job variable cannot be read (possibly no read authorization or missing password).

Result:

The system is not operated by the run control system; any tasks that exist are assigned the status ERROR.

- The MSCF job variable can be read with the status value (position 1 and 2) = \$R.

Result:

The run control system accepts the system as an active system and the MSCF job variable is mapped to ≠ \$R with an event query (INACTIVE control).

- The MSCF job variable can be read with the status value (position 1 and 2) <> \$R.

Result:

The run control system accepts the system as an inactive system and the MSCF job variable is mapped to \$R with an event query (ACTIVE control).

When the run control system is started, all possible systems are monitored and the monitoring results are logged.

The following requirements must be satisfied if jobs are to be started and executed on other BS2000 systems:

- The home pubset of the system must be available (status REMOTE-HOME shown with /SHOW-MASTER-CATALOG-ENTRY) and
- MSCF must display the status "running" for the system (value for \$TSOS.SYS.MSCF.CONTROL-STATE.<system> = \$R).

Jobs that are scheduled to start during the time a system is down are assigned the status HOSTWAIT. These jobs are started again automatically by AVAS as soon as the system is again available. The run control system automatically recognizes when this is the case, and jobs with the status HOSTWAIT are started immediately for the system involved. From this point on, job distribution is again running normally.

AVAS monitors the "running" status of all other BS2000 systems by means of an event query as to whether the value of the corresponding system job variable is equal to or not equal to \$R.

Specifications in the net and job descriptions

Through the statements for processing the net and job descriptions (CREATE-NET-DESCRIPTION, MODIFY-NET-DESCRIPTION and SHOW-NET-DESCRIPTION), the NET-CAT (mask AVN001) and JOB-CAT (mask AVN002) parameters are available to the user in order to address the target processor for execution of the nets and jobs. Here the user can enter either a catalog ID or the name of a job variable. The job variable (JV) must be supplied with the catalog ID by the user at the time of net release (SUBMIT); the JV is evaluated at this time by the AVAS function (AVAS-internal GETJV macro call). The AVAS function must be able to access the JV; any existing JV password must be made known through the start procedure.

The catalog ID *ANY is interpreted by the run control system and is converted to a catalog ID using the BS2000 JMS (Job Management System). The task is then started using the catalog ID established in this way.

*ANY can only be used in an XCS network.

Tasks which are defined in the net description with JOB-TYPE=EXT and are to be brought to execution on another processor must be created by the user in such a way that they can be accessed by the host processor.

The SHOW-NET-STATUS statement can be used to display the NET-CAT (mask AVI002) and JOB-CAT (mask AVI003) parameters. The MODIFY-SUBMIT-NET statement (masks AVD001, AVD002) is used to modify them.

The PARTNER-NAME parameter (CREATE-/MODIFY-/SHOW-NET-DESCRIPTION, mask AVN016) is provided for addressing FT remote hosts. These hosts are administered completely within the *openFT* subsystem. The names of the FT remote hosts must be agreed on accordingly. The SERVER monitor of AVAS does not take the statuses of these hosts into account.

Processing of the nets and tasks by AVAS

The catalog ID of the shared pubset or SPD on which AVAS is to create the ENTER files is made known to AVAS by means of the SPVS or SPD parameter in the start procedure of the run control system. When working with a shared pubset, the user ID of the AVAS run control system must be entered in the user catalog of the shared pubset to allow AVAS to create the ENTER files.

On the ENTER call, the monitoring job variables are created by AVAS on the relevant home pubset. To permit this, the user ID of the AVAS run control system must also exist on the AVAS slave processors. For the monitoring of all tasks on the slave processors, the run control system links an event with the modification of the task job variable, for which it waits for \$T or \$A in the first two positions.

If no ENTER call can be issued because access to an AVAS slave processor is not possible, the job is placed in ERROR CAT status (displayed with SHOW-NET-STATUS, mask AVI013).

7.2 Controlling remote systems with AVAS-SV-BS2

7.2.1 Controlling and monitoring of server jobs

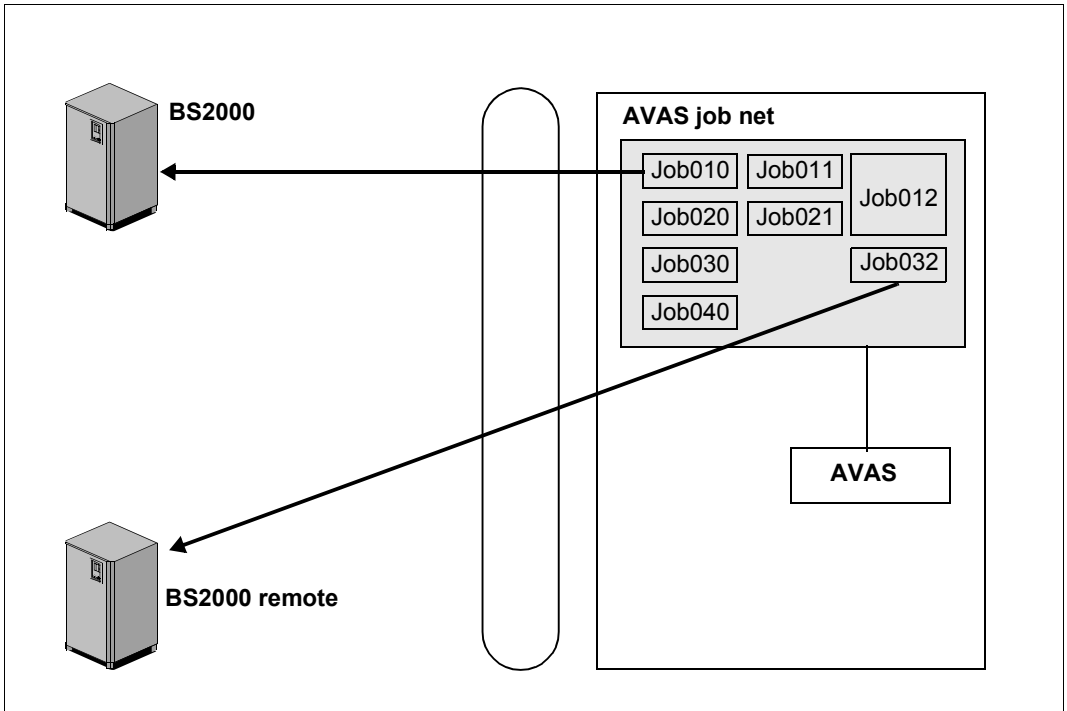


Figure 14: Cross-platform job scheduling

With an AVAS job net it is possible to control jobs which are dependent on each other which run on different systems: BS2000 jobs for local and remote BS2000 systems. The jobs are started centrally by AVAS, monitored, and administered centrally with their logs.

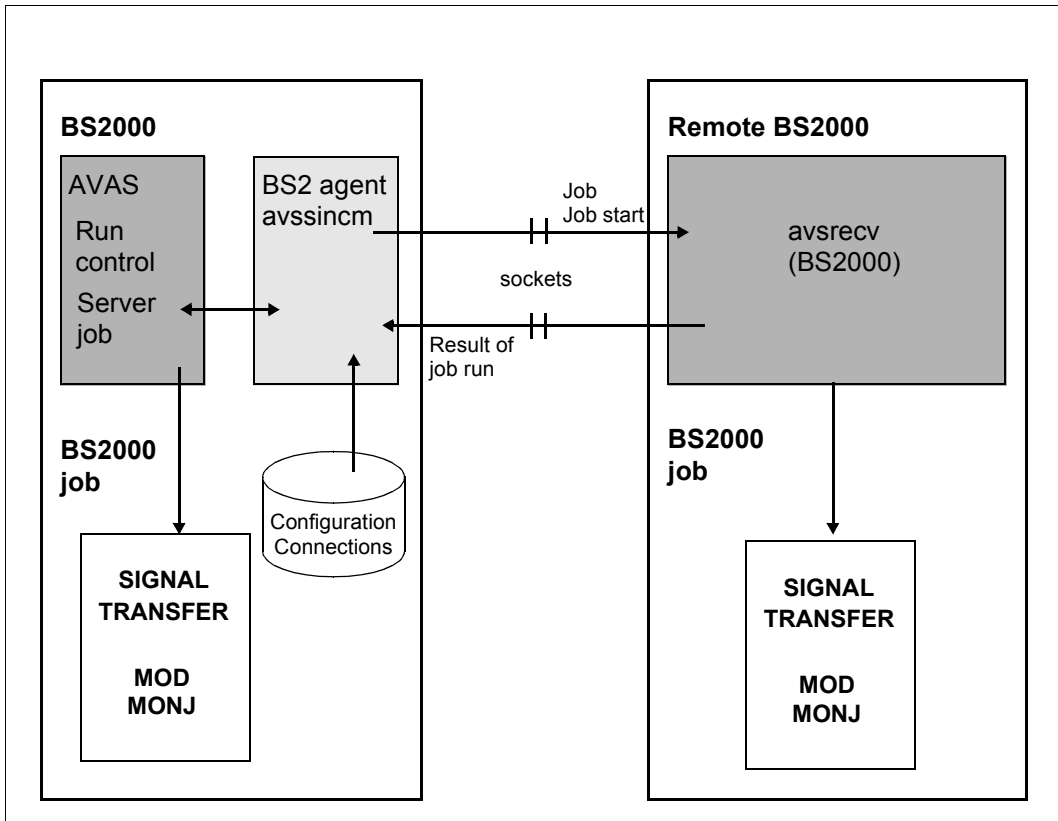


Figure 15: Cross-platform control and monitoring for remote BS2000 systems

AVAS also starts jobs on remote BS2000 systems. Communication between the local and remote systems takes place via the socket interface, which means that MSCF is not required for these remote BS2000 systems. The server agent `avsrecv` is installed on the remote system.

The connection and authorization data of the remote BS2000 is entered in the configuration file on the local system. Consequently a corresponding server job variable must be configured for each remote BS2000 server. The server monitor then places the current status of the BS2000 server in this.

For the server jobs which are to be started the BS2000 agent requires a server name which is used to ascertain the target system from the configuration file. This server name must be entered in the net or job description in place of the catalog ID. The jobs themselves can be started either locally or remotely without any modification.

Runtime log files are sent directly from the remote jobs to the DCAM application CENTRAL and are stored by this in the log library. Transfer takes place using the SIGNAL and TRANSFER programs which are integrated in the jobs.

Entries in the monitoring job variable are returned with the result of the job run and processed by the run control system. With regard to the job status, this processing takes place as for jobs started locally.

7.2.2 Monitoring of the remote systems/servers

To ensure that server jobs are only started on active servers, AVAS provides a monitoring function for the servers entered in the configuration file. This is implemented with the server monitoring process.

Server jobs which are to be run on a server which is not currently available are given the status HOSTWAIT.

A job variable with the name AVS.SERVER.STATE.<server-jv-link-name>.<server-name> must be created under the BS2000 ID of the run control system for each server entered in the configuration file. This AVAS SERVER job variable is used by the run control system to check the availability of the server. Updating of the SERVER monitor job variable is implemented by the AVAS server monitoring process.

The run control system sets the status of server jobs when they are started depending on the first two positions of the SV-JV:

RUNNING	If the value in both positions of the SV-JV is '\$R'
HOSTWAIT	As long as the values in the first two positions of the SV-JV are not '\$R'
ERROR	When there is no corresponding SV-JV. In this case the server job cannot be run.

Functions of the AVAS server monitor process

Determining the server from the configuration file

When the server monitor is started, the configuration files are read and the SV-JVs are set accordingly. The server monitor creates SV-JVs if there aren't any already for a server to be monitored.

After that, the state of the AVAS server is determined. The server monitor sends messages to the server to be monitored and waits for its replies.

If the corresponding server sends a reply, the value '\$R' is entered in the associated SV-JV instead of '\$U'. If no reply is received, then the SV-JV is set to the value '\$T'.

Cyclical server check

The servers are checked when server monitoring is started. This check is performed cyclically in an interval set with the SV-CONTROL-TIME operand.

When servers are added to or removed from the configuration file, issuing the CONFIG operation to the server monitoring process updates the server environment to be monitored.

The run control system must be informed of the change. This is done either with the `/INFORM-PROGRAM MSG='USERVER',JOB-ID=*TSN(<tsn>)` command or by assigning `USERVER` to the run time control system's job variables.

The following entries in the SV-JV are evaluated by the run control system:

- \$U** Undefined: The state of the server is unknown .
Jobs for this host are always started.
- \$R** Running: The server is ready for operation and can accept jobs.
Jobs for this host are placed in the `RUNNING` state.
- \$T** Terminated: The `AVSSINCM` program could not establish a connection to the server. The server is not ready for operation or cannot be reached via the computer network. Jobs cannot be accepted.
Jobs for this host are placed in the `HOSTWAIT` state.
- \$D** Deleted: The corresponding server is not entered in the configuration file.
Jobs for this host are placed in the `ERROR` state.
- \$I** Ignore: The server is currently not being monitored. The status must be changed manually.
Jobs for this host are placed in the `HOSTWAIT` state.

How to temporarily lock a server

Job distribution by the run control system can be disabled for a short time for a server in order to configure the server, for example. The corresponding server job variable is set to `$I` (Ignore) to accomplish this.

If the lock is to be removed, set the value back to `'$R'` (`RUNNING`).

7.2.3 Tools for displaying and managing decentralized systems/servers

The following tools are available for displaying the status information of and managing the AVAS servers supported by AVAS.

BS2000

The AVS.SVSTATUS procedure in the SYSPRC.AVAS.085 library shows the status information of the AVAS servers. Here the procedure parameter INFORMATION controls the scope of the information which is displayed:

- INFORMATION=*STATUS displays the status information of the servers.
- INFORMATION=*VERSION displays the status information and the version of the AVAS server.

The AVS.EDITCONFIG procedure permits the configuration file to be edited simply in EDT.

Browser interfaces

The information on the AVAS server systems can be displayed using any Web browser.

To permit this, the server monitor process must be started in BS2000. The server interface process which handles communication with the browser must also be started. The AVSSURF program makes the server interface process available.

The server interface process enables the AVAS server systems contained in the configuration file and the associated status information to be displayed both in a general view and in a server-specific detailed view. In the case of a server, the AVAS jobs running on it and their job data (status, start time, process ID, signaled log files) can be displayed. Furthermore, the configuration file can be edited and the status of the servers can be modified (in the SERVER job variables).

Some of these functions are offered only when the user has the appropriate authorization. For this purpose the user can authenticate himself/herself by specifying the AVAS system ID, the AVAS user ID and the associated password. To permit authentication the server interface process forwards this data to the AVAS system together with the specified system ID.

Communication between the browser and the server interface process can optionally be encrypted.

8 Automation of net runs

A high degree of automation is only feasible if recurrent functions can be implemented on the procedural level. AVAS supports this by offering batch statements for the majority of AVAS action functions.

8.1 BATCH statements

A fully or partially qualified net name and the time interval can be specified for BATCH statements referring to nets.

The following batch statements are available:

ADD-JOB-LOG	Add log data
CANCEL-NET	Interrupt or abort a net owing to an error
CHANGE-NET-DESCRIPTION	Change an entire net description
COPY-ELEMENT	Copy library elements
CREATE-PLAN-NET	Plan net processing
CREATE-PERIOD	Create a period
CREATE-PROD-JOB	Create static jobs and S procedures
CREATE-PROD-NET	Create the temporary jobs and S procedures of a net
DELETE-DOCUMENT	Delete documentation elements
DELETE-JOB	Delete jobs, JCL elements and S procedures
DELETE-JOB-LOG	Delete logs of a net
DELETE-NET-DESCRIPTION	Delete nets
DELETE-PERIOD	Delete a period
DELETE-PLAN-NET	Delete planned nets from the production plan
DELETE-PROD-JOB	Delete static jobs and S procedures
END	Terminate the stream of batch statements
HOLD-NET	Hold nets currently being executed

REPEAT-NET	Repeated release of planned net
RESTART-NET	Restart a net after an error
RESUME-NET	Cancel the hold status
SIGNON	Start the stream of batch statements
SUBMIT-NET	Release planned nets

8.2 AVAS program interface

Selected AVAS action functions are also provided on a program interface (Assembler, COBOL). These functions are:

- SIGNON
- CREATE-PLAN-NET
- CREATE-PROD-NET
- SUBMIT-NET
- RESTART-NET
- END

Read access to the run control and journal file is also possible via a program interface. The following data can be obtained:

- net overview, net structures
- journal net overview, journal data of a net
- overview of condition descriptions and data for a condition.

9 Administration

Authorization concept

Authorization on the underlying system

AVAS does not require any special privileges (BS2000 user ID TSOS). Faults or operator errors therefore do not affect the underlying BS2000 system.

Access control

The entire job production is protected by personalized access control via AVAS user IDs and passwords. AVAS users can be mutually authorized for certain functions or data.

Function-oriented authorization

Each AVAS user can only invoke those functions for which he or she is authorized. Thus, the staff involved in creating, planning, scheduling, handling, and monitoring production can be clearly distinguished.

Data-oriented authorization

A number of AVAS users can be combined to form a user group. Each user can only access the net and job libraries assigned to his or her user group. This allows the nets and jobs of different departments to be made mutually inaccessible.

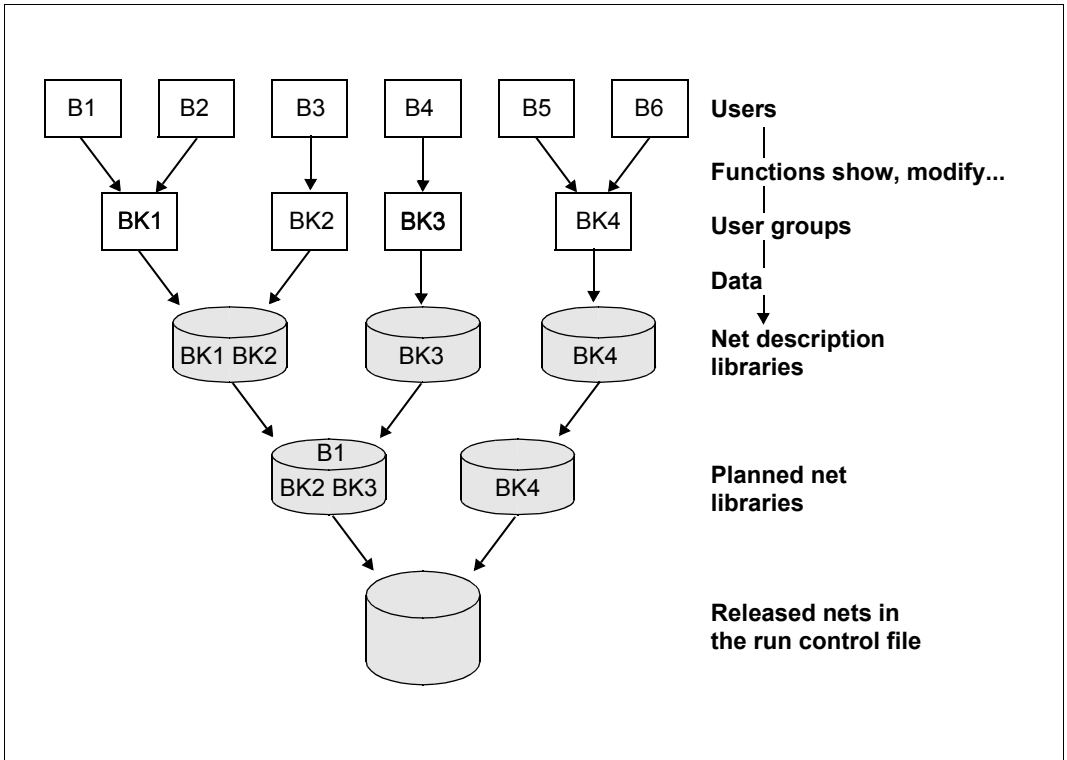


Figure 16: Function and data-oriented authorization in AVAS

Task-oriented authorization

By combining function and data-oriented authorization, each user group can be restricted to their own particular tasks, e.g. department-specific creation of nets/jobs and cross-department production scheduling.

Function authorizations

Since the objects in the files (NETLIB, JCLLIB, DOCLIB, NPRLIB, JMDLIB, ABLDAT etc.) are addressed via the user group, a user normally exercises his function authorization on the objects of the user group assigned to him.

Moreover, the user can also be given a privileged authorization if he is granted the function authorization for all elements in the file assigned to his user group.

The elements in a function authorization table are formed with the aid of statements of the AVAS system.

Each table element is assigned an entry. This entry can have any of three statuses:

- (0): no authorization for the statement
- (1): user-group-related authorization for the statement (privileged user). All elements of the user group assigned to the user can be processed.
- (*): authorization for the statement which is not specific to user groups (privileged user). All elements in the file assigned via the user group can be processed. Elements of user groups working with other files cannot be processed.

Since two or more users can work with the same function authorization table, direct changes in the tables are not permitted during the current session. However, the user can be assigned another function authorization table which is already known to the AVAS system. This table does not come into effect until the relevant user has signed on to AVAS again with SIGNON. It is not possible to create new elements for a foreign user group (CREATE ... statements).

The function authorizations of the user with regard to the AVAS system are handled by the AVAS administrator. There must be at least one user defined as the AVAS administrator. AVAS administrators are defined by the fact that they are granted authorization for the SHOW-SYSTEM-PARAMS and MODIFY-SYSTEM-PARAMS statements. These statements can be used to display or modify the AVAS system parameters. The function authorizations of a user with regard to the AVAS system are stored in the system parameters.

User administration and communication

For user communication, AVAS offers functions for informing and displaying the users currently logged on. The AVAS administrator has control over all users. If required, he or she can deliberately terminate or block other users' activities.

The following statements are available for administration and communication. They are described in the "AVAS Statements" [1] manual:

CANCEL-USER	Forcibly sign off users
SEND-MESSAGE	Forcibly sign off users
SHOW-USER	Forcibly sign off users

System settings

The installation-specific configuration of the AVAS system is defined using AVAS system parameters. These include, for example, default values for most parameters in the net description, the AVAS functions, and specifications on AVAS behavior when processing a restart.

AVAS exits

At various points in the production run, AVAS provides exit interfaces that allow the system to branch to a data center routine, where it is possible to modify or extend the sequence of AVAS processing for that particular installation.

CC exits when saving nets and jobs

When nets and jobs are stored, user-specific version controls for the various elements can be configured via CC exits.

CC exits in production preparation

The option of predefining values in the input masks for production preparation and checking the values entered for plausibility using CC-defined routines not only contributes to the automation process but also considerably improves the quality of the production preparation process.

CC exits during release for production

The release of a net can be monitored, logged, permitted, or rejected in data center routines. Statements or parameters can be inserted, modified or deleted in the jobs.

CC exit while starting a job

A job can be processed in a data center routine before being started by AVAS. Data such as passwords can thus be queried or modified immediately before the job is started. This exit is not activated for FT requests.

CC exit while terminating a job or net

As soon as a job or an AVAS net terminates normally or abnormally, AVAS can branch to a data center routine in which the AVAS system administrator can provide error recovery sequences, particularly for unmanned data center environments. In the event of the abnormal termination of strategic jobs, it is thus possible to call a standby service or, in the case of dependent procedures, a replacement strategy.

CC exit during journal output

All actions are logged in the journal. Therefore, the user cannot only adapt the journal output at this exit but also monitor the entire process.

Reorganization

AVAS files must be reorganized at regular intervals. This involves removing processed nets from the job and journal files. If desired, the data in the libraries for the production plan (NPRLIB and JMDLIB) and LOGSYS can also be reorganized. Journal records and log entries can be saved during reorganization.

10 Tables and overviews

10.1 AVAS statements and parameters

The statements are alphabetically sorted by the objects (column 2 of the following table) which the statements are applied to.

Statement		Parameters
COPY	CALENDAR	[CALENDAR-NAME=calendar]
CREATE		CALENDAR-NAME=calendar
DELETE		[CALENDAR-NAME=calendar]
MODIFY		[CALENDAR-NAME=calendar] [,SYMDAT-NAME=symdat] [,PERIOD-NAME=period / (dd.mm.yy[,dd.mm.yy])]
SHOW		[CALENDAR-NAME=calendar] [,PERIOD-NAME=period / (dd.mm.yy[,dd.mm.yy])]
ADD	COND-DESCRIPTION	CONDITION-NAME=[\$ug_]condname [,TYPE=RES / VAL]
DELETE		[CONDITION-NAME=[\$ug_]condname] [,TYPE=NET / JOB / RES / VAL] [,STATUS=ABENDED / CREATED / DELETED / ENDED / ERROR / EXCLUSIVE / FREE / IGNORED / NO-PLAN / NO-SUBMIT / SHARE / SKIPPED]
MODIFY		[CONDITION-NAME=[\$ug_]condname] [,TYPE=NET / JOB / RES / VAL] [,OBJECT= <u>DES</u> / USR] [,STATUS=ABENDED / CREATED / DELETED / ENDED / ERROR / EXCLUSIVE / FREE / IGNORED / NO-PLAN / NO-SUBMIT / SHARE / SKIPPED]
SHOW		[CONDITION-NAME=[\$ug_]condname] [,TYPE=NET / JOB / RES / VAL] [,OBJECT= <u>DES</u> / USR] [,STATUS=ABENDED / CREATED / DELETED / ENDED / ERROR / EXCLUSIVE / FREE / IGNORED / NO-PLAN / NO-SUBMIT / SHARE / SKIPPED]

Statement		Parameters
DELETE	DOCUMENT	[ELEMENT-NAME={\$ug_element}]
EDIT		[ELEMENT-NAME={\$ug_element}]
SHOW		[ELEMENT-NAME={\$ug_element}]
COPY	ELEMENT	MODE=LIBOUT / LIBIN[,USER-GROUP={\$ug} / SAMOUT / SAMIN
SHOW	FORMAT	[FORMAT-NAME=format] [,AVAS-USER-LIBRARY= <u>NETMAP</u> / JOBMAP]
DELETE	JOB	[ELEMENT-NAME={\$ug_element}]
EDIT		[ELEMENT-NAME={\$ug_element}]
SHOW		[ELEMENT-NAME={\$ug_element}]
MODIFY-SUBMIT		[NET-NAME={\$ug_netname}] [,OBJECT=STR] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=YES / NO]
ADD	JOB-LOG	[NET-NAME={\$ug_netname}]
DELETE		[NET-NAME={\$ug_netname}]
SHOW		[NET-NAME={\$ug_netname}]
SHOW	JOURNAL	[NET-NAME={\$bk_netname}] [PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,RUN-CONTROL-SYSTEM=*STD / avak]
SEND	MESSAGE	[USER-NAME=name] [,USER-GROUP=\$ug / *ALL]
CANCEL	NET	[NET-NAME={\$ug_netname}] [,CANCEL-TYPE=SOFT / HARD] [PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,RUN-CONTROL-SYSTEM=*STD / avak] KILL-JOBS= <u>NO</u> /YES
HOLD		[NET-NAME={\$ug_netname}] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,RUN-CONTROL-SYSTEM=*STD / avak]

Statement		Parameters
NET-CONTROL	NET (cont.)	[NET-NAME=[<u>\$ug_</u> netname] [,OBJECT=NET / STR] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,NET-STATUS=ABENDED / ENDED / ERROR / RUNNING / HOLD / OPWAIT / START / RESUMED / WAITING / CONDWAIT / HOSTWAIT / RESTARTED / MODIFIED / IGNORED / SHIFTED] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=NO / YES]
REPEAT		[,NET-NAME=[<u>\$ug_</u> netname] [,OBJECT= <u>NET</u> / STR] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,DISPLAY=NO / YES]
RESTART		[NET-NAME=[<u>\$ug_</u> netname] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=NO / <u>YES</u>]
RESUME		[NET-NAME=[<u>\$ug_</u> netname] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,RUN-CONTROL-SYSTEM=*STD / avak]
START		[NET-NAME=[<u>\$ug_</u> netname] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,NET-STATUS= <u>OPWAIT</u> / WAITING / ALL] [,RUN-CONTROL-SYSTEM=*STD / avak]
SUBMIT		[,NET-NAME=[<u>\$ug_</u> netname] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,OBJECT= <u>NET</u> / STR] [,DISPLAY=NO / YES]

Statement		Parameters
CHANGE	NET- DESCRIPTION	[NET-NAME=[\${ug}_]netname]
COPY		[NET-NAME=[\${ug}_]netname]
CREATE		NET-NAME=[\${ug}_]netname [,OBJECT=NET / PST / MAP / STR]
DELETE		[NET-NAME=[\${ug}_]netname]
MODIFY		[NET-NAME=[\${ug}_]netname] [,OBJECT=NET / PST / MAP / STR]
SHOW		[NET-NAME=[\${ug}_]netname] [,OBJECT=NET / PST / MAP / STR]
COLLECT	NET-PARAMS	[NET-NAME=[\${ug}_]netname] [,FORMAT-NAME=format]
SHOW	NET-STATUS	[NET-NAME=[\${ug}_]netname] [,OBJECT=NET / STR] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]])] [,NET-STATUS=ABENDED / CONDWAIT / ENDED / ERROR / HOLD / HOSTWAIT / IGNORED / MODIFIED / NETWAIT / OPWAIT / RESTARTED / RESUMED / RUNNING / SHIFTED / START / WAITING] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=YES / NO]
CREATE	ORDER	[NET-NAME=[\${ug}_]netname]
CREATE	PERIOD	PERIOD-NAME=period
DELETE		[PERIOD-NAME=period]
MODIFY		[PERIOD-NAME=period]
SHOW		[PERIOD-NAME=period]

Statement		Parameters
CREATE	PLAN-NET	[PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]]) [,CALENDAR-NAME=calendar / *STD] [,NET-NAME=[\$ug_]netname] [,DISPLAY=NO / YES] [,ALTERN-NET-NAME=[\$ug_]netname]
DELETE		[PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]]) [,NET-NAME=[\$ug_]netname] [,NET-STATUS=TOCREATE / PARTIALLY / CREATED / NOTTOCREATE / SUBMITTED / REPEATED] [,DISPLAY=ALL]
MODIFY		[PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]]) [,NET-NAME=[\$ug_]netname]
SHOW		[PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]]) [,NET-NAME=[\$ug_]netname] [,NET-STATUS=TOCREATE / PARTIALLY / CREATED / NOTTOCREATE / READYFORSUBM / SUBMITTED / REPEATED] [,DISPLAY=NO / YES]
CREATE	PROD-JOB	[INPUT-NAME=[\$ug_]jobname] [,OUTPUT-NAME=[\$ug_]jobname] [,FORMAT-NAME=format]
DELETE		[ELEMENT-NAME=[\$ug_]element]
EDIT		[ELEMENT-NAME=[\$ug_]element]
SHOW		[ELEMENT-NAME=[\$ug_]element]
CREATE	PROD-NET	[NET-NAME=[\$ug_]netname] [,PERIOD-NAME=period / (dd.mm.yy[/hh:mm:ss][,dd.mm.yy[/hh:mm:ss]]) [,DISPLAY=NO / YES]
DELETE		[NET-NAME=[\$ug_]netname [,NET-STATUS=PARTIALLY / CREATED]]
MODIFY		[NET-NAME=[\$ug_]netname]
MODIFY	SUBMIT-JOB	[NET-NAME=[\$ug_]netname] [,OBJECT=STR] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=NO / YES]

Statement		Parameters
MODIFY	SUBMIT-NET	[NET-NAME=[\$ug_]netname] [,OBJECT=NET / STR] [,RUN-CONTROL-SYSTEM=*STD / avak] [,DISPLAY=NO / YES]
COPY	SYSTEM-ELEMENT	AVAS-USER-LIBRARY=NETLIB / JCLLIB / JMDLIB / DOCLIB [,ELEMENT-NAME=[\$ug_]element]
DELETE		AVAS-SYSTEM-LIBRARY=NETSYS / JCLSYS / JMDSYS / DOCSYS [,ELEMENT-NAME=[\$ugsys_]element]
MODIFY	SYSTEM-PARAMS	[RECORD=keyword]
SHOW		[RECORD=keyword]
CANCEL	USER	[USER-NAME=name] [,USER-GROUP=\$ug / *ALL] [,CANCEL-TYPE=SOFT / HARD]
SHOW		[USER-NAME=name] [,USER-GROUP=\$ug / *ALL]

Statements without object

Statement	Parameters
EDT	
START-EXIT	
START-MONITOR	[NET-NAME=[\$ug_]netname] [,RUN-CONTROL-SYSTEM=*STD / avak]

10.2 AVAS statements via /INFORM-PROGRAM

10.2.1 Operating the run control system

Command	Parameters
/INFORM-PROGRAM	MSG='CANCEL',JOB-ID=*TSN(<tsn>)
	MSG='HOLD[,LEVEL= <u>NET</u> / JOB]',JOB-ID=*TSN(<tsn>)
	MSG='NETC',JOB-ID=*TSN(<tsn>)
	MSG='RESUME',JOB-ID=*TSN(<tsn>)
	MSG='STOP[,LEVEL= <u>NET</u> / JOB]',JOB-ID=*TSN(<tsn>)
	MSG='RUNC',JOB-ID=*TSN(<tsn>)
	MSG='UHOST',JOB-ID=*TSN(<tsn>)
	MSG='USERVER',JOB-ID=*TSN(<tsn>)

10.2.2 Controlling released nets via calling the run control system

Command	Parameters
/INFORM-PROGRAM	MSG='CANCEL-NET,NET-NAME=netname [,CANCEL-TYPE= <u>SOFT</u> / HARD] [,KILL-JOBS= <u>NO</u> / YES] ,JOB-ID=*TSN(<tsn>)
	MSG='HOLD-NET,NET-NAME=netname' ,JOB-ID=*TSN(<tsn>)
	MSG='RESTART-NET,NET-NAME=netname [,RESTART-VARIANT=1 / 2 / 3] [,ERROR-INDEX=index] [,ERROR-NAME=name] ,JOB-ID=*TSN(<tsn>)
	MSG='RESUME-NET,NET-NAME=netname' ,JOB-ID=*TSN(<tsn>)
	MSG='SHOW-NET-STATUS[,NET-NAME=\$bk_[netname]]' ,JOB-ID=*TSN(<tsn>)
	MSG='START-NET,NET-NAME=netname' ,JOB-ID=*TSN(<tsn>)

10.2.3 Access to the SYSOUT/SYSLST of the AVAS processes

Command	Parameters
/INFORM-PROGRAM	MSG='COPYLST',JOB-ID=*TSN(<tsn>)
	MSG='COPYOUT',JOB-ID=*TSN(<tsn>)
	MSG='NEWLST',JOB-ID=*TSN(<tsn>)
	MSG='NEWOUT',JOB-ID=*TSN(<tsn>)

10.3 AVAS operations and operation characters

The operation names and characters can be input in the CMD: mask field or via function keys.

CMD:+,-,+n,-n,F[IRST],L[AST],++,--

In all lists (masks marked on the bottom right with an asterisk (*)), the operation codes specified above can be used for scrolling.

CMD:E As a rule, element processing is begun by marking the desired elements in the masks with the overview of elements ('List' column) and entering CMD:E. The marks are then processed.

If overviews of records are displayed in element masks (e.g. AVN004 - net structure), the individual records can likewise be processed by marking and then entering CMD:E.

The marks have the following meanings:

Y The selected element is processed immediately (e.g. COPY-CALENDAR).

S The selected element is displayed for processing by the user (e.g. MODIFY-NET-DESCRIPTION).

N The marked element is not processed.

CMD:R This terminates element processing immediately.;

CMD:R can be used in any mask.

CMD:R in masks with the overview of elements ('List' column) causes the command function to be aborted.

CMD:R in masks where elements are displayed ('Element' column) causes element processing to be aborted. AVAS starts processing the next element or, if there are no more marked elements, returns to the element overview.

CMD:S This stores the element.

CMD:CO This means that the entries made in this mask are fixed and the next marked element is displayed.

CMD:I This means that the entries made in this mask are ignored and the next marked element is displayed.

CMD:CH This calls the CHECK function.

CMD:J This displays a job log.

CMD:P This outputs the data to a print file.

CMD:D	This displays the documentation which is stored for the displayed object (net, job or condition) in the AVAS system.
CMD:?	This outputs information on the statement specified in the CMD field. If a mask for the statement already exists, the permissible entries are described. Otherwise the parameters will be described.
CMD:#1n	<p>This group of operations causes the operations "Save", "Return" etc. to be executed.</p> <p>The following individual operations are possible:</p> <ul style="list-style-type: none">#11 corresponds to EXECUTE#12 corresponds to SAVE#13 corresponds to CONTINUE#14 corresponds to RETURN#15 corresponds to IGNORE#16 corresponds to CHECK#17 corresponds to DOCUMENT#18 corresponds to PRINT#19 corresponds to JOBLOG
CMD:#2n	<p>This group of operations branches to NET processing.</p> <p>The following individual operations are possible:</p> <ul style="list-style-type: none">#21 calls the HOLD-NET function#22 calls the RESUME-NET function#23 calls the CANCEL-NET function#24 calls the RESTART-NET function#25 calls the START-NET function#26 calls the MODIFY-SUBMIT-NET function
CMD:#3n	<p>This group of operations branches to JOB processing.</p> <p>The following individual operations are possible:</p> <ul style="list-style-type: none">#31 calls the MODIFY-SUBMIT-JOB function#32 calls the SHOW-SUBMIT-JOB function#33 calls the NET-CONTROL function for a subnet

CMD:#4n	This group of operations branches to CONDITION processing. The following individual operations are possible: #41 calls the ADD-CONDITION-DESCRIPTION function #42 calls the MODIFY-CONDITION-DESCRIPTION function #43 calls the SHOW-CONDITION-DESCRIPTION function #44 calls the SHOW-NET-STATUS function
CMD:#5n	This group of operations branches to JOURNAL and JOBLOG processing. The following individual operations are possible: #51 calls the SHOW-JOURNAL function #52 calls the SHOW-HISTORY function #53 calls the ADD-JOB-LOG function #54 calls the SHOW-JOB-LOG function #55 calls the START-EXIT function
CMD:#6n	This group of operations branches to the functions for net planning and release. The following individual operations are possible: #61 calls the CREATE-PLAN-NET function #62 calls the CREATE-PROD-NET function #63 calls the SUBMIT-NET function
CMD: #7n	This group performs operations on BS2000 objects. #71 Operation displays/blanks out BS2000 passwords #72 call BS2000 command SHOW-JOB-STATUS or SHOW-FILE-TRANSFER or SHOW-JV #73 calls the BS2000 command CANCEL-JOB or CANCEL-FILE-TRANSFER #74 calls the BS2000 command MODIFY-JV #75 calls the BS2000 command INFORM-JOB #76 calls the BS2000 command INFORM-PROG #77 calls the BS2000 command HOLD-TASK #78 calls the BS2000 command RESUME-TASK #79 displays the SYSOUT file of the AVAS job

Notes

- You can also specify the name of the operation instead of the operation number, e.g.:
#EXECUTE instead of #11
#START-NET instead of #25
#SHOW-NET-STATUS instead of #44
- A #operation or #function cannot be specified in conjunction with the F or K key.

10.4 AVAS statements and mask sequences

10.4.1 Overview of AVAS statements and mask sequences

The statements are alphabetically sorted by the objects which the statements are applied to.

***-notation:**

You can page in all lists (mask is marked with '*' on the bottom right) with the specified operations.

Statements	List	Element	Record
SIGNON		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS010: Signon data </div> <p style="text-align: center;">↓ CMD:L</p>	
??		<p style="text-align: center;">↓</p> <div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS020: Overview of functions * </div> <p style="text-align: center;">CMD:E</p>	

Statements		List	Element	Record
COPY	CALENDAR	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC012: List of calendars * </div> <p style="text-align: center;">CMD:E</p>		
CREATE			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC001: List of FREE days * </div> <p style="text-align: center;">CMD:S</p>	
DELETE		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC010: List of calendars * </div> <p style="text-align: center;">CMD:E</p>		
MODIFY			<pre> graph LR A["AVC010: List of calendars *"] -- "CMD:E" --> B["AVC002: Calendar data *"] B -- "CMD:E, S" --> C["EDT mask: List of symdats *"] C -- "EDT:H" --> D["AVC004: Return from EDT *"] D -- "CMD:CO,I" --> B </pre>	
SHOW			<pre> graph LR A["AVC010: List of calendars *"] -- "CMD:E" --> B["AVC002: Calendar data *"] B -- "CMD:R" --> C["EDT mask: List of Symdats *"] </pre>	

Statements		List	Element	Record	
ADD	COND-DESCRIPTION		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD030: Condition description </div>	CMD:S,D	
DELETE		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD040: Condition entries * </div>		CMD:E	
MODIFY		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD040: Condition entries * </div>	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVD030: Condition description </div> <div style="border: 1px solid black; padding: 5px;"> AVD031: Condition user * </div> </div>	CMD:CO,S,D	CMD:E,CO
SHOW		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD040: Condition entries * </div>	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVD030: Condition description * </div> <div style="border: 1px solid black; padding: 5px;"> AVD031: Condition user * </div> </div>	CMD:CO,D	CMD:E,CO

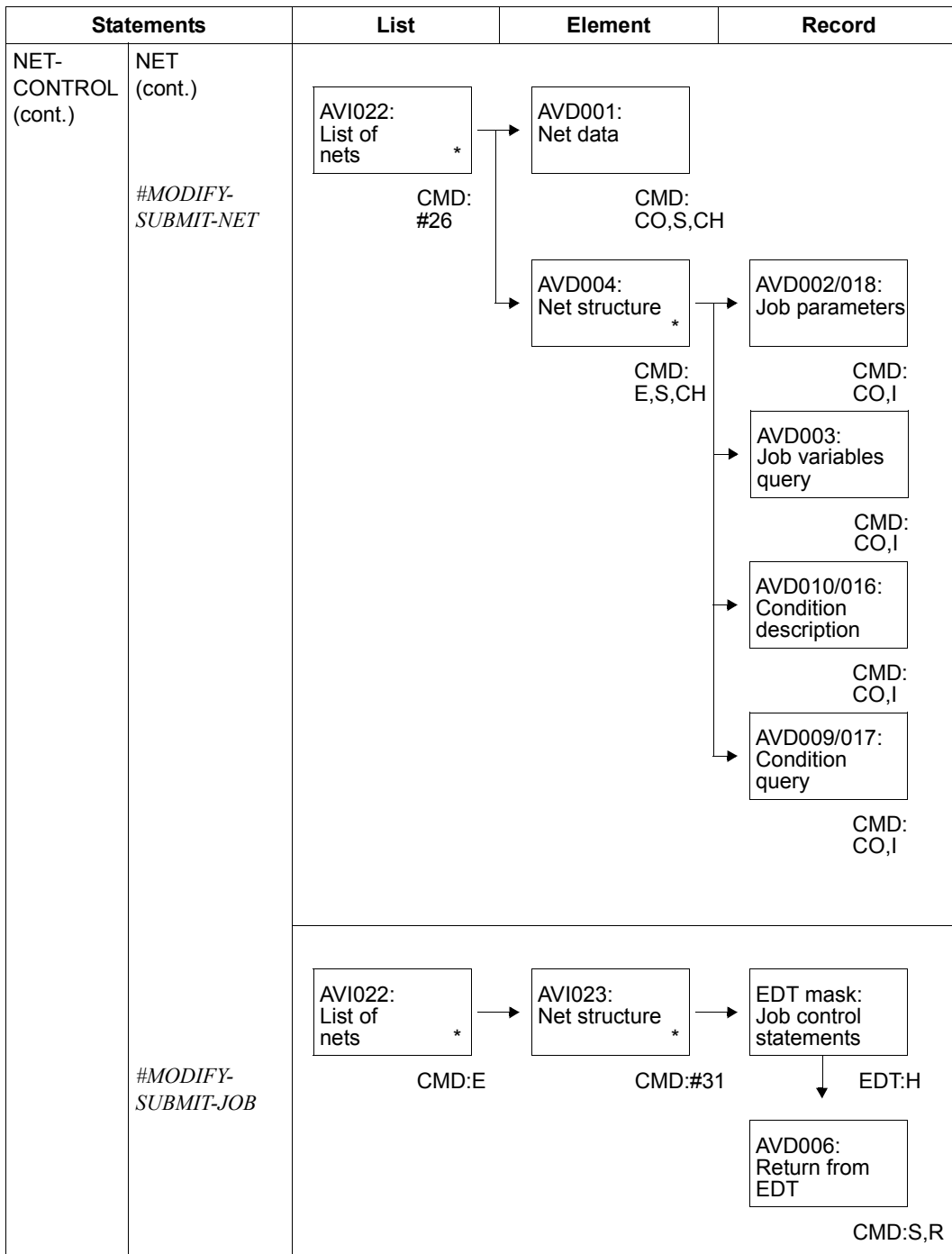
Statements		List	Element	Record
DELETE	DOCUMENT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS019: List of documents * </div> <p style="text-align: center;">CMD:E</p>		
EDIT	DOCUMENT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS019: List of documents * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> EDT mask: Document data records </div> <p style="text-align: center;">EDT:H</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS016: Return from EDT </div> <p style="text-align: center;">CMD:S,R</p>
SHOW	DOCUMENT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS019: List of documents * </div> <p style="text-align: center;">CMD:E,P</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> EDT mask: Document data records </div> <p style="text-align: center;">EDT:H</p>	
COPY	ELEMENT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS011: List of library elements* </div> <p style="text-align: center;">CMD:E</p>		
SHOW	FORMAT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI015: List of formats * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> User mask: Fields for modification </div> <p style="text-align: center;">CMD:R</p>	

Statements		List	Element	Record
DELETE	JOB	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E</p>		
EDIT		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> EDT mask: Job control statements </div> <p style="text-align: center;">EDT:H</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE011: return from EDT </div> <p style="text-align: center;">CMD:S,R</p>
SHOW		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E,P</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> EDT mask: Job control statements </div> <p style="text-align: center;">EDT:H</p>	
ADD	JOB-LOG	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI016: List of nets * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI017: List of job runs * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI018: Logs of job execution * </div> <p style="text-align: center;">CMD:CO,E</p>
				<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI019: Logs of job execution </div> <p style="text-align: center;">CMD:CO,I</p>

Statements	List	Element	Record
DELETE	JOB-LOG (cont.)	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI016: List of nets * </div> <p style="text-align: center;">↓ CMD:E</p> <div style="border: 1px solid black; padding: 5px;"> AVI017: List of job runs * </div> <p style="text-align: center;">CMD:E</p>	
SHOW		<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI016: List of nets * </div> <p style="text-align: center;">↓ CMD:E,P</p> <div style="display: flex; justify-content: space-between;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI017: List of job runs * </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI018: Logs of job execution * </div> </div> <p style="text-align: center;">CMD:E,P CMD:CO,E,P</p> <div style="border: 1px solid black; padding: 5px; margin-left: auto;"> EDT mask: Log data </div> <p style="text-align: right;">EDT:H</p>	
SHOW	JOURNAL	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI014: List of nets * </div> <p style="text-align: center;">CMD:E,P</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI005: AVAS actions * </div> <p style="text-align: center;">CMD:E,P,J</p> <div style="border: 1px solid black; padding: 5px;"> AVI006: Single information </div> <p style="text-align: center;">CMD: CO,I,P,J</p> <p>CMD:J displays the logs via EDT.</p>	

Statements	List	Element	Record
SEND	MESSAGE	<p>AVS035: List of users *</p> <p>CMD:E,I</p> <p>AVS036: Mask for message *</p> <p>CMD:E</p>	<p>CMD:I means that the mask is supplied with the new current values.</p>
CANCEL	NET	<p>AVD015: List of nets *</p> <p>CMD:E</p>	
HOLD		<p>AVD015: List of nets *</p> <p>CMD:E</p> <p>AVD008: Structure description *</p> <p>CMD:E</p>	
NET-CONTROL		<p>Standardnets:</p> <p>AVI022: List of nets *</p> <p>CMD:E</p> <p>AVI023: Structure description *</p> <p>CMD:#nn,CO</p> <p>AVI002: Net parameters</p> <p>CMD:CO</p> <p>AVDnnn: Structure description *</p> <p>Hypernets/Subnets:</p> <p>AVI022: List of nets *</p> <p>CMD:E</p> <p>AVI023: Structure of hypernet *</p> <p>CMD:E</p> <p>AVI023: Net parameters of subnet *</p> <p>CMD:CO</p> <p>AVI0nn: Structure description *</p>	

Statements		List	Element	Record
NET-CONTROL (cont.)	NET (cont.)			
	<i>#HOLD-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI022: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD008: Structure description * </div> <p style="text-align: center;">CMD:#21 CMD:E</p>		
	<i>#RESUME-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI022: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD008: Structure description * </div> <p style="text-align: center;">CMD:#22 CMD:E</p>		
	<i>#CANCEL-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI022: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD008: Structure description * </div> <p style="text-align: center;">CMD:#23 CMD:E</p>		
	<i>#RESTART-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI022: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD007: Net structure * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD005: Restart jobs * </div> <p style="text-align: center;">CMD:#24 CMD:E,S,D CMD:CO,I,D</p>		
<i>#START-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI022: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVD008: Structure description * </div> <p style="text-align: center;">CMD:#25 CMD:E</p>			



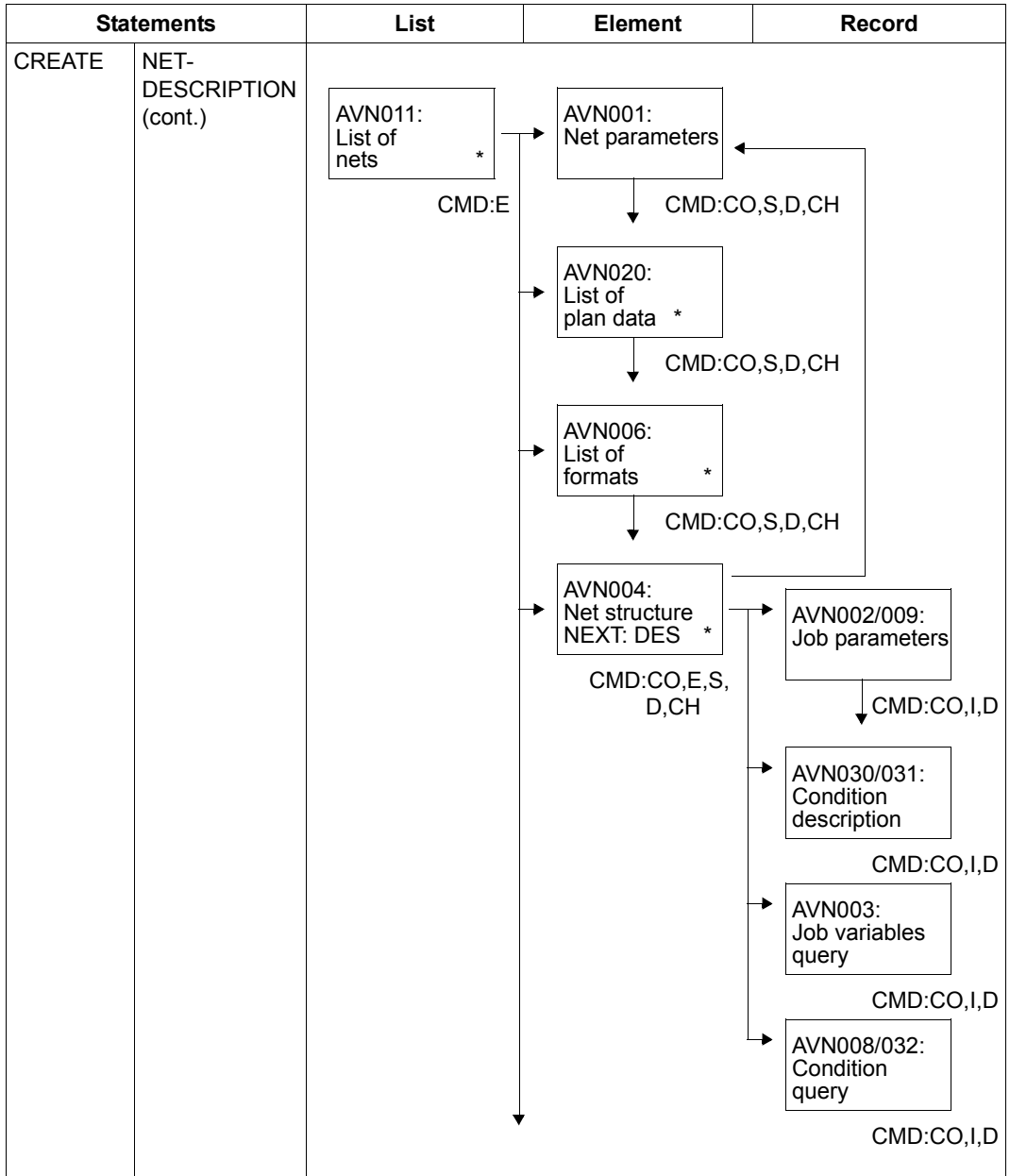
Statements		List	Element	Record
NET-CONTROL (cont.)	NET (cont.)	AVI022: List of nets *	AVI023: Net structure *	EDT mask: Job control statements
	<i>#SHOW-SUBMIT-JOB</i>	CMD:E	CMD:#32	EDT:H
	<i>#NET-CONTROL</i>	AVI022: List of nets *	AVI023: Net structure Hypernet *	
		CMD:E	CMD:#33	
			AVI023: Net structure Subnet *	
			CMD:#nn	
			AVnnn: Net data Subnet	

Statements	List	Element	Record
<p>NET-CONTROL (cont.)</p> <p>NET (cont.)</p> <p><i>#ADD-CONDITION-DESCRIPTION</i></p> <p><i>#MODIFY-CONDITION-DESCRIPTION</i></p> <p><i>#SHOW-CONDITION-DESCRIPTION</i></p> <p><i>#SHOW-NET-STATUS</i></p>	<p>AVI022: List of nets *</p>	<p>AVI023: Net structure *</p> <p>AVI023: Net structure *</p> <p>AVI023: Net structure *</p> <p>AVI023: Net structure *</p>	<p>AVD030: Condition description</p> <p>AVD030: Condition description</p> <p>AVD030: Condition description</p> <p>AVD030: Condition description</p>
<p><i>#SHOW-JOURNAL</i></p>	<p>AVI022: List of nets *</p>	<p>AVI005: AVAS action *</p>	<p>AVI006: Single Information</p>
<p><i>#SHOW-HISTORY</i></p>	<p>AVI022: List of nets *</p>	<p>AVI035: History data</p>	

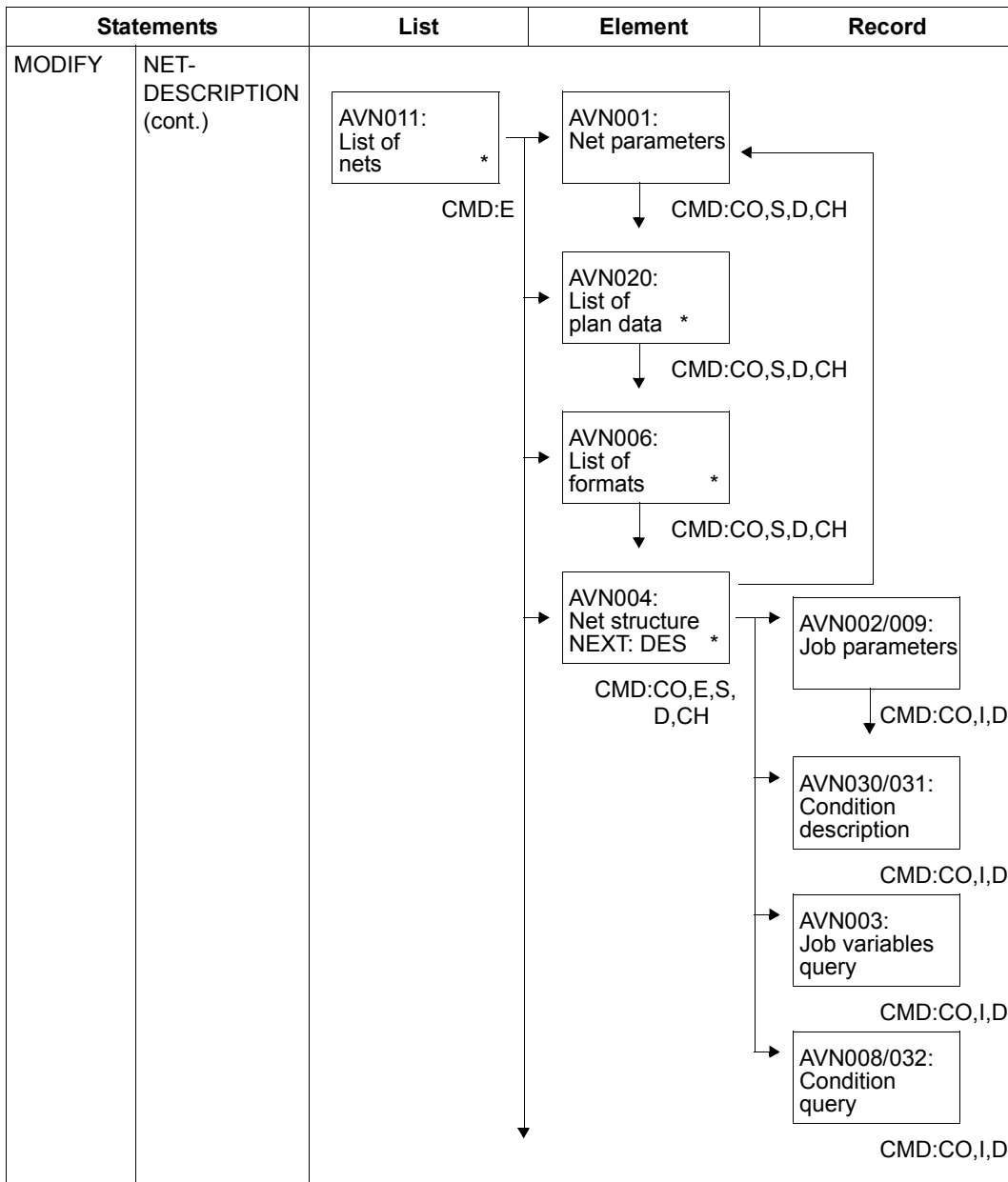
Statements	List	Element	Record
<p>NET-CONTROL (cont.)</p> <p>NET (cont.)</p> <p><i>#ADD-JOB-LOG</i></p>	<p>AVI022: List of nets *</p> <p>↓</p> <p>AVI017: List of job runs *</p> <p>↓</p> <p>AVI018: Logs of job execution *</p> <p>↓</p> <p>AVI019: Log of job execution</p>	<p>CMD:#54</p> <p>EDT:H</p> <p>CMD:E,P</p> <p>CMD:CO,E</p>	<p>EDT:H</p> <p>EDT:H</p>
<p><i>#SHOW-JOB-LOG</i></p>	<p>AVI022: List of nets *</p> <p>↓</p> <p>AVI017: List of job runs *</p> <p>↓</p> <p>EDT mask: Log data</p> <p>↓</p> <p>EDT mask: Log data</p>	<p>CMD:#54</p> <p>EDT:H</p> <p>CMD:E,P</p> <p>CMD:CO,E,P</p>	<p>EDT:H</p> <p>EDT:H</p>

Statements	List	Element	Record
REPEAT NET (cont.)		<pre> graph LR A["AVF001: List of nets *"] -- "CMD:E" --> B["AVF012: Net data *"] A -- "CMD:E" --> C["AVF014: Net structure *"] B --> L1["List"] C --> L1 B -- "CMD:S,CO" --> R1["Record"] C -- "CMD:S,CO" --> R1 </pre>	
RESTART		<pre> graph LR A["AVD012: List of nets *"] -- "CMD:E" --> B["AVD007: Net structure *"] B -- "CMD:E,S,R,D" --> C["AVD005: Restart jobs *"] B --> L1["List"] C --> R1["Record"] </pre>	
RESUME		<pre> graph LR A["AVD015: List of nets *"] -- "CMD:E" --> B["AVD008: List of structure description *"] A --> L1["List"] B --> R1["Record"] </pre>	
START		<pre> graph LR A["AVD015: List of nets *"] --> L1["List"] A -- "CMD:E" --> R1["Record"] </pre>	
SUBMIT		<pre> graph LR A["AVF001: List of nets *"] -- "CMD:E" --> B["AVF002: Net data *"] A -- "CMD:E" --> C["AVF004: Net structure *"] B --> L1["List"] C --> L1 B -- "CMD:S,CO" --> R1["Record"] C -- "CMD:S,CO" --> R1 </pre>	

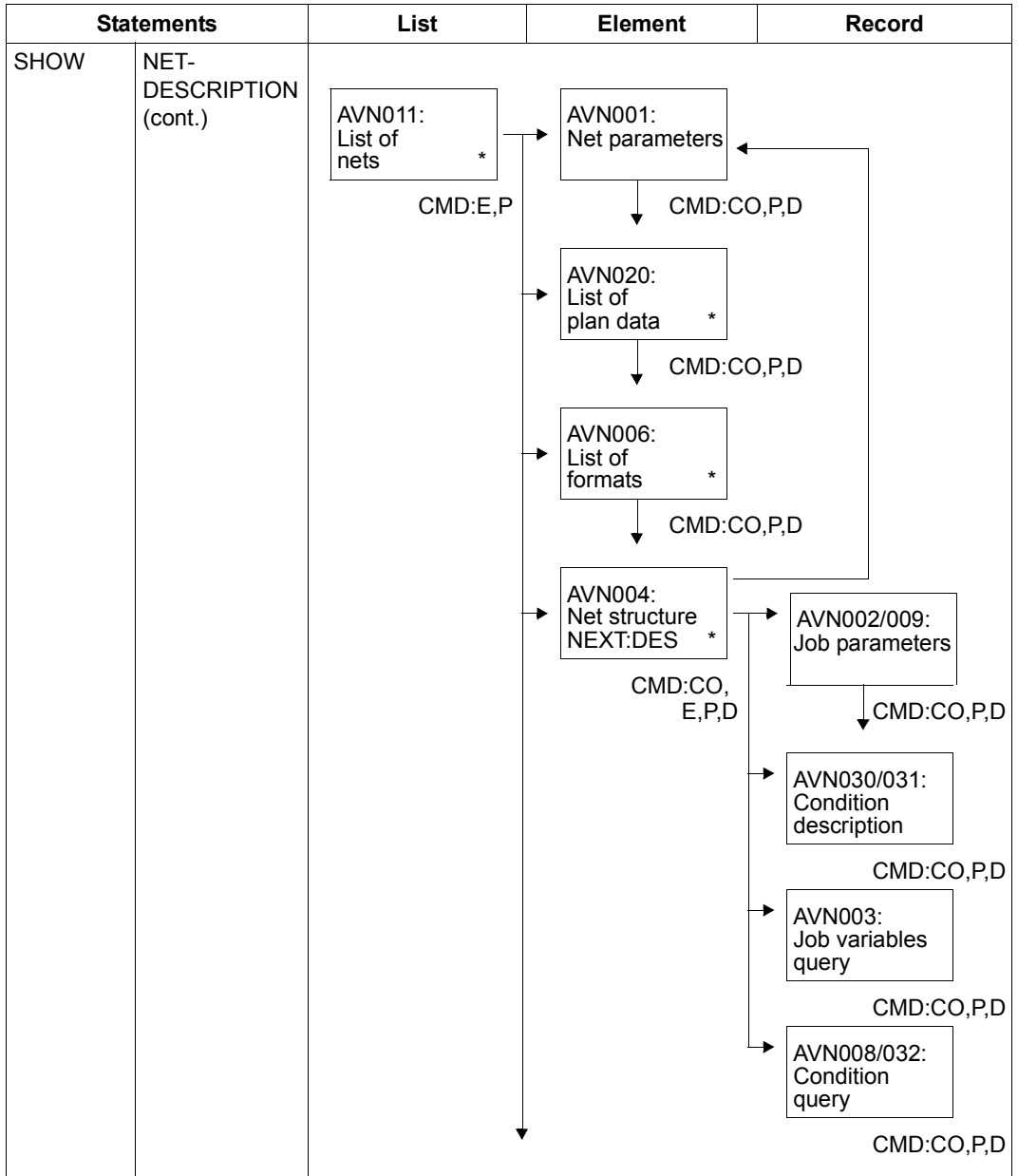
Statements		List	Element	Record
CHANGE	NET- DESCRIPTION	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVN011: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVN007: Net parameters * </div>
		CMD:E		CMD:CO
COPY		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVN012: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVN005: Net structure * </div>
		CMD:E		CMD:S



Statements		List	Element	Record
CREATE (cont.)	NET- DESCRIPTION (cont.)		<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVN004: Net structure NEXT: SYM * CMD:E,S,D,CH </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVN004: Net structure NEXT: JCL * CMD:E </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> EDT mask: Job control statements EDT:H </div> <div style="border: 1px solid black; padding: 5px;"> AVE011: Return from EDT * CMD:S,R </div>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVN021: Plan data jobs * CMD:CO,I,D </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVN022/023: Plan data cond. query * CMD:CO,I,D </div> <div style="border: 1px solid black; padding: 5px;"> AVN024: Plan data cond. descr. * CMD:CO,I,D </div>
DELETE			<div style="border: 1px solid black; padding: 5px;"> AVN011: List of nets * CMD:E </div>	



Statements		List	Element	Record
MODIFY (cont.)	NET- DESCRIPTION (cont.)		<pre> graph TD A["AVN004: Net structure NEXT: SYM * CMD:E,S,D,CH"] --> B["AVN021: Plan data * CMD:CO,I,D"] B --> C["AVN022/023: Plan data cond. query * CMD:CO,I,D"] B --> D["AVN024: Plan data cond. descr. * CMD:CO,I,D"] A --> E["AVN004: Net structure NEXT: JCL * CMD:E"] E --> F["EDT mask: Job control statements EDT:H"] F --> G["AVE011: Return from EDT * CMD:S,R"] </pre>	



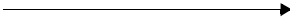



Statements		List	Element	Record
SHOW (cont.)	NET- DESCRIPTION (cont.)		<pre> graph TD A["AVN004: Net structure NEXT: SYM *"] -- "CMD:E,P,D" --> B["AVN021: Plan data jobs *"] B -- "CMD:CO,P,D" --> C["AVN022/023: Plan data cond. query *"] B -- "CMD:CO,P,D" --> D["AVN024: Plan data cond. descr. *"] E["AVN004: Net structure NEXT: JCL *"] -- "CMD:E" --> F["EDT mask: Job control statements"] </pre>	
COLLECT	NET- PARAMS		<pre> graph LR A["AVM010: List of nets *"] -- "CMD:E" --> B["AVM011: List of user masks *"] B -- "CMD:E,S" --> C["User mask: Values for modification"] </pre>	

Statements		List	Element	Record
SHOW	NET-STATUS	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI012: List of nets * </div> <p style="margin-left: 40px;">CMD: E,I,P</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI002: Net parameters </div> <p style="margin-left: 40px;">CMD: CO,I,P,D</p>	
			<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI013: Net structure * </div> <p style="margin-left: 40px;">CMD: E,I,P,D,J</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI003/020: Job parameters </div> <p style="margin-left: 40px;">CMD: CO,I,P,D,J</p>
				<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI004: Job variables query </div> <p style="margin-left: 40px;">CMD: CO,I,P,D</p>
				<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI007/010: Condition query </div> <p style="margin-left: 40px;">CMD: CO,I,P,D</p>
				<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI008/009: Condition descriptions </div> <p style="margin-left: 40px;">CMD: CO,I,P,D</p>
				<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> AVI025: Parameter structure elem. Fu=s </div> <p style="margin-left: 40px;">CMD: CO,I,P,D</p>
		<p>CMD:I means that the mask is supplied with the new current values.</p>		

Statements		List	Element	Record
CREATE	ORDER	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>Mark. Y/N CMD:E</p>		
		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>Mark. S CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP001: Net structure * </div>	
		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>CMD:#61</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP001: Net structure * </div> <p>CMD:S</p>	
		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>CMD:#62</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVM001: Net structure * </div> <p>CMD:S</p>	
		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>CMD:#52</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI035: HISTORY data </div> <p>CMD:R</p>	
		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVP012: List of nets * </div> <p>CMD:#51</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVI005: AVAS functions * </div> <p>CMD:R</p>	
	#CREATE-PLAN-NET			
	#CREATE-PROD-NET			
	#SHOW-HISTORY			
	#SHOW-JOURNAL			

Statements		List	Element	Record
CREATE (cont.)	ORDER (cont.) <i>#SUBMIT-NET</i>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVP012: List of nets * </div> CMD:#63	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVF002: Net data </div> CMD:S	
CREATE	PERIOD		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC020: Period data </div> CMD:S	
DELETE		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC021: List of periods * </div> CMD:E		
MODIFY SHOW		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC021: List of periods * </div> CMD:E	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVC020: Period data </div> CMD:S	

Statements		List	Element	Record
CREATE	PLAN-NET	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVP011: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVP001: Net structure * </div>		
		CMD:E,P	CMD:S,P	
DELETE		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVP010: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVP003: Net data </div>		
		CMD:E	CMD:E	
MODIFY		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVP010: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVP003: Net data </div>		
		CMD:E	CMD:S	
SHOW		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVI011: List of nets * </div> → <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-left: 20px;"> AVI001: Net processing status * </div>		
		CMD:E,P	CMD:R,P	

Statements		List	Element	Record
CREATE	PROD-JOB	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVM013: List of jobs * </div> <p style="text-align: center;">CMD:E</p>	 <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 200px;"> User mask: Values for modification </div>	<p>CMD:CO,I</p>
DELETE		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E</p>		
EDIT		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E</p>	 <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 20px;"> EDT mask: Job control statements </div> <p style="text-align: right; margin-right: 20px;">EDT:H</p>  <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 100px;"> AVE011: Return from EDT </div> <p style="text-align: center;">CMD:S,R</p>	
SHOW		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVE010: List of jobs * </div> <p style="text-align: center;">CMD:E,P</p>	 <div style="border: 1px solid black; padding: 5px; width: fit-content; margin-left: 20px;"> EDT mask: Job control statements </div> <p style="text-align: right;">EDT:H</p>	

Statements		List	Element	Record				
CREATE	PROD-NET	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM012: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM001: Job records of the net * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> User mask: Values for modification </div>	CMD:E CMD:E,S CMD:CO,I	
DELETE		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM020: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM001: Job records of the net * </div>				CMD:E CMD:E
MODIFY		<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM020: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVM001: Job records of the net * </div>				CMD:E CMD:E
MODIFY	SUBMIT-JOB	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD011: List of nets * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD004: Net structure * </div>	→	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> EDT mask: Job control statements </div>	CMD:E CMD:E,R EDT:H	
						↓	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVD006: Return from EDT </div>	CMD:S,R

Statements		List	Element	Record
MODIFY	SUBMIT-NET	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD011: List of nets * </div> <p style="text-align: center;">CMD:E</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD001: Net data </div> <p style="text-align: center;">CMD:CO,S,CH</p>	
			<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD004: Net structure * </div> <p style="text-align: center;">CMD:E,S,CH</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD002/018: Job parameters </div> <p style="text-align: center;">CMD:CO,I</p>
				<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD003: Job variables query </div> <p style="text-align: center;">CMD:CO,I</p>
				<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD010/016: Condition description </div> <p style="text-align: center;">CMD:CO,I</p>
				<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVD009/017: Condition query </div> <p style="text-align: center;">CMD:CO,I</p>
COPY	SYSTEM-ELEMENT	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS012: List of elements * </div> <p style="text-align: center;">CMD:E</p>		
DELETE		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS013: List of elements * </div> <p style="text-align: center;">CMD:E</p>		

Statements		List	Element	Record
MODIFY	SYSTEM-PARAMS	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS001: Parameter groups * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS002: Files * </div>	
			CMD:E CMD:E,S	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS003: Users * </div>	
			CMD:E,S	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS004: User groups * </div>	
			CMD:E,S	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS005: Function table * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS005: CMD authorization * </div>
			CMD:E	CMD:E,S
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS006: File allocation * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS006: Individual allocation * </div>
			CMD:E	CMD:E,S
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS007: Run control systems * </div>	
			CMD:E,S	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS008: S#nnn variable * </div>	
			CMD:E,S	

Statements		List	Element	Record
SHOW	SYSTEM-PARAMS (cont.)	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS001: Parameter- groups * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS002: Files * </div>	
		CMD:E	CMD:R	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS003: Users * </div>	
			CMD:R	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS004: User groups * </div>	
			CMD:R	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS005: Function table * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS005: CMD authorization * </div>
			CMD:E	CMD:R
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS006: File allocation * </div>	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS006: Individual allocation * </div>
			CMD:E	CMD:R
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS007: Run control systems * </div>	
			CMD:R	
			<div style="border: 1px solid black; padding: 5px; display: inline-block;"> AVS008: S#nnn- variable * </div>	
			CMD:R	

Statements		List	Element	Record
CANCEL	USER	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS035: List of users * </div> <p style="text-align: center;">CMD:E,I</p> <p>CMD:I means that the mask is supplied with the new current values.</p>		
SHOW		<div style="border: 1px solid black; padding: 5px; width: fit-content;"> AVS035: List of users * </div> <p style="text-align: center;">CMD:I</p>		

10.4.2 Representation of the mask sequences with NET-CONTROL

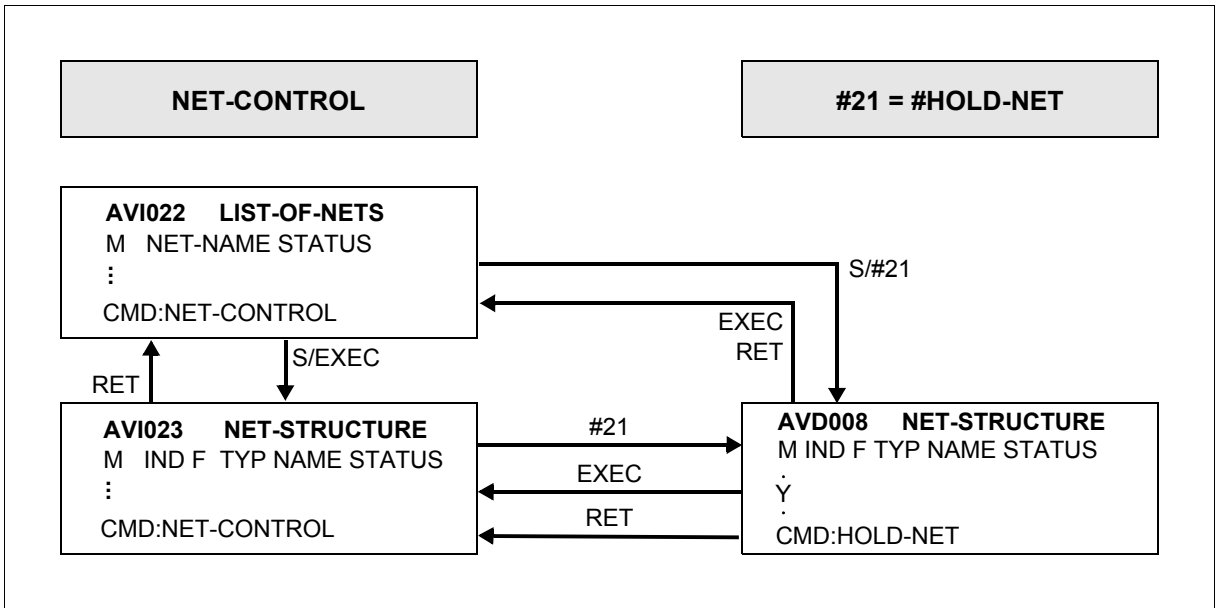


Figure 17: NET-CONTROL, CMD = #21 = #HOLD-NET

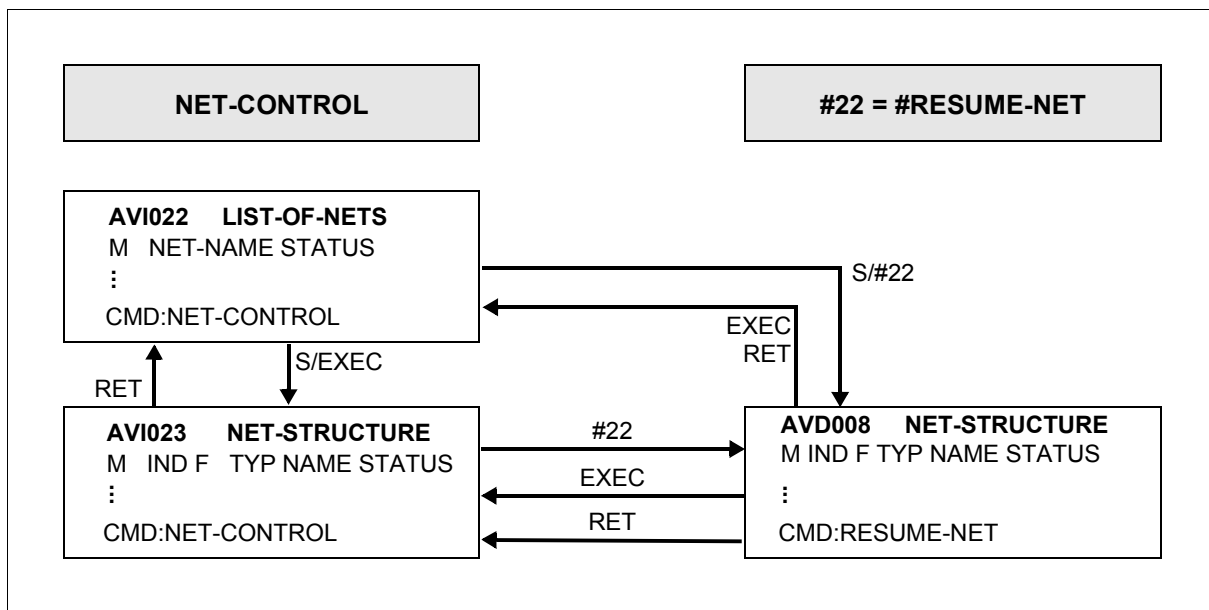


Figure 18: NET-CONTROL, CMD = #22 = #RESUME-NET

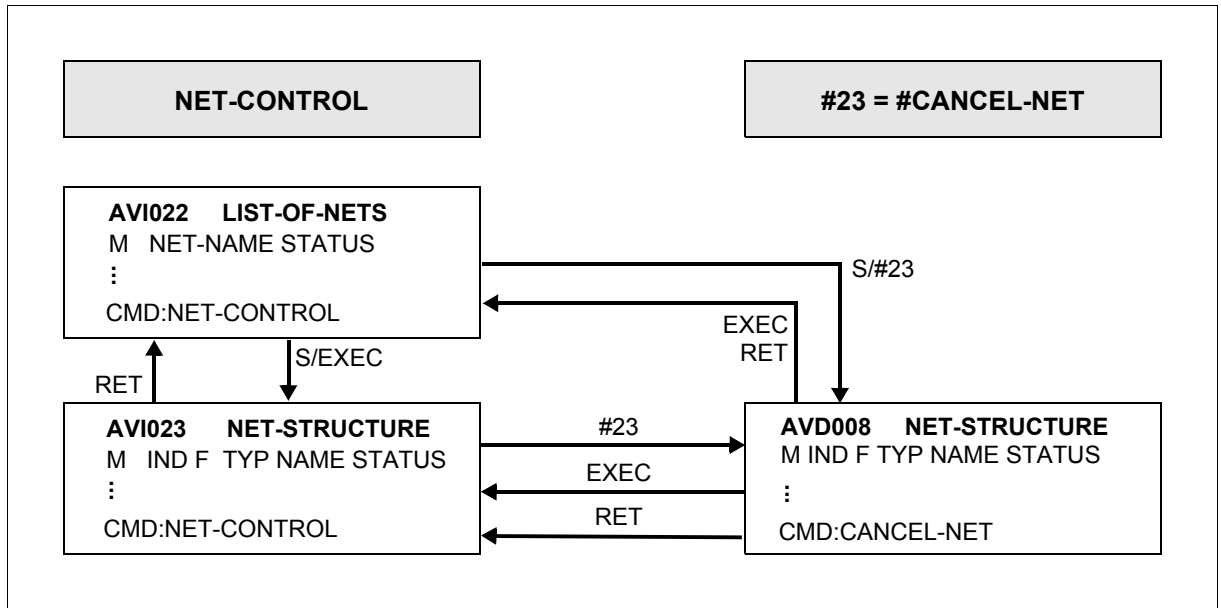


Figure 19: NET-CONTROL, CMD = #23 = #CANCEL-NET

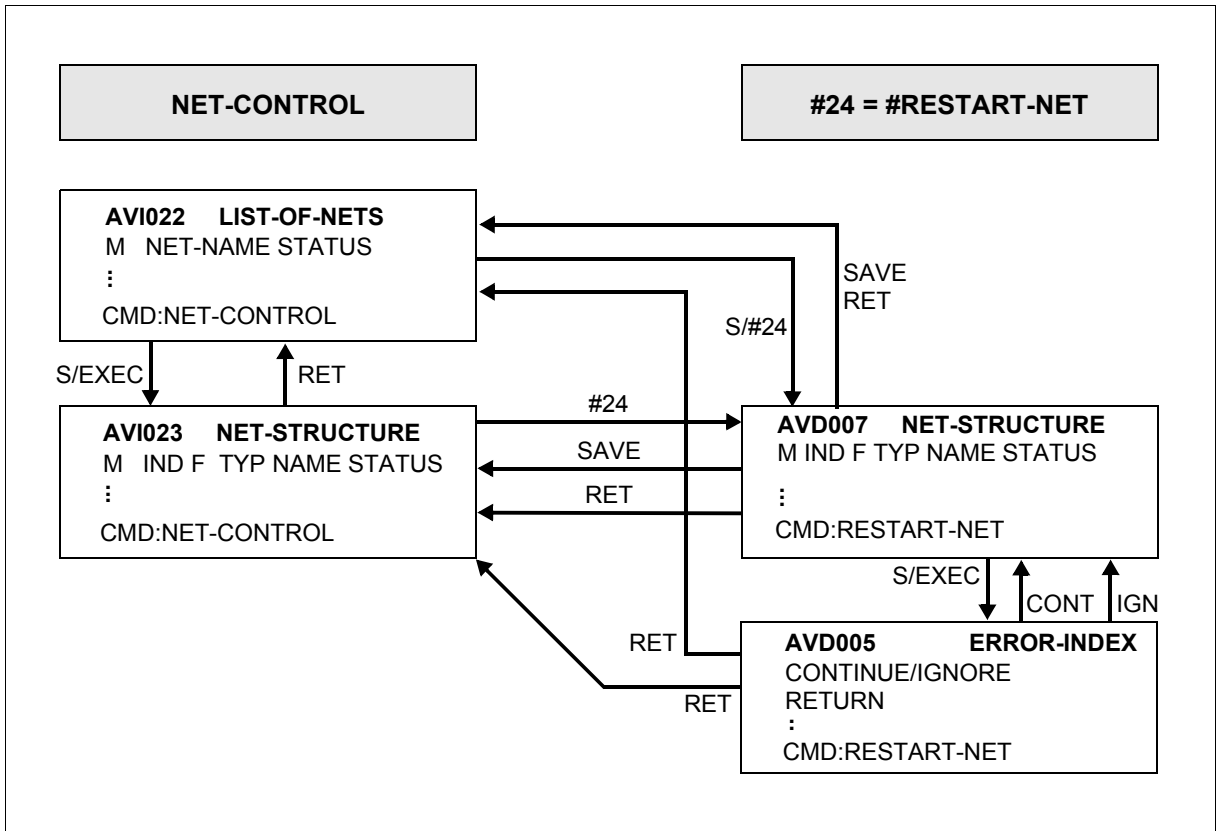


Figure 20: NET-CONTROL, CMD = #24 = #RESTART-NET

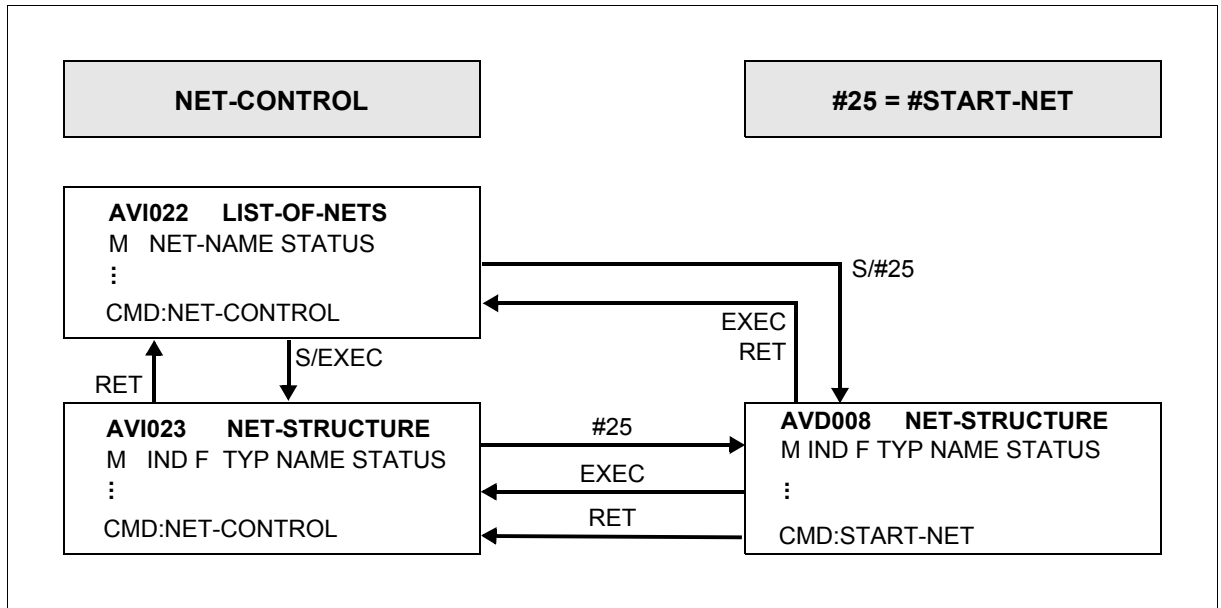


Figure 21: NET-CONTROL, CMD = #25 = #START-NET

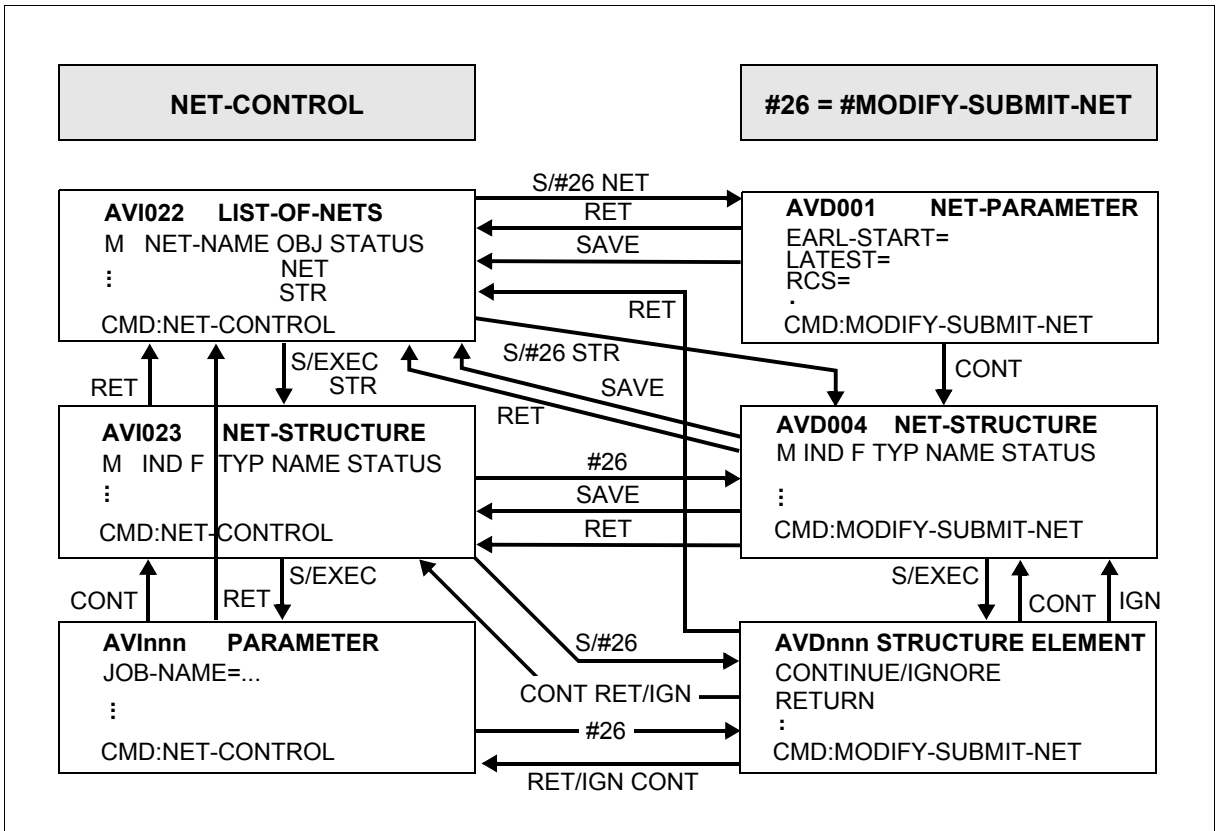


Figure 22: NET-CONTROL, CMD = #26 = #MODIFY-SUBMIT-NET

Note on AVInnn and AVDnnn

Mask sequence for modifying the structure elements:

Mask name	CMD	Mask name	Mask content
AVI003	#26	AVD002	Structure element with FU=J / P
AVI004		AVD003	Structure element with FU=C (JVA)
AVI007		AVD009	Structure element with FU=C (NET / JOB / RES / VAL)
AVI008		AVD010	Structure element with FU=A / M / D (RES / VAL)
AVI009		AVD016	Structure element with FU=D (NET / JOB)
AVI010		AVD017	Structure element with FU=W (TIM)
AVI026		AVD026	Structure element with FU=F(TRA)

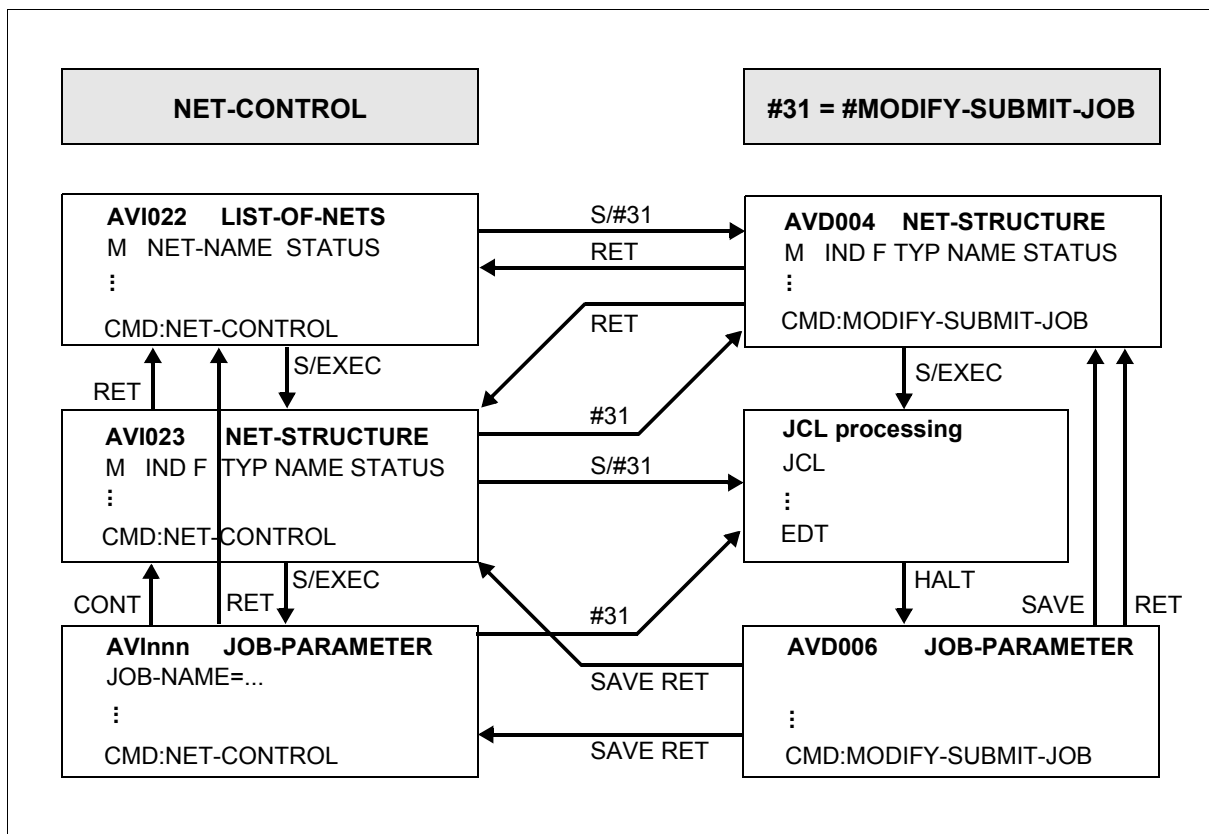


Figure 23: NET-CONTROL, CMD = #31 = #MODIFY-SUBMIT-JOB

Note on AVInn

- In the case of structure elements with FU=J or FU=P mask AVI003 is displayed.
- In the case of a structure element with FU=F mask AVI026 is displayed.

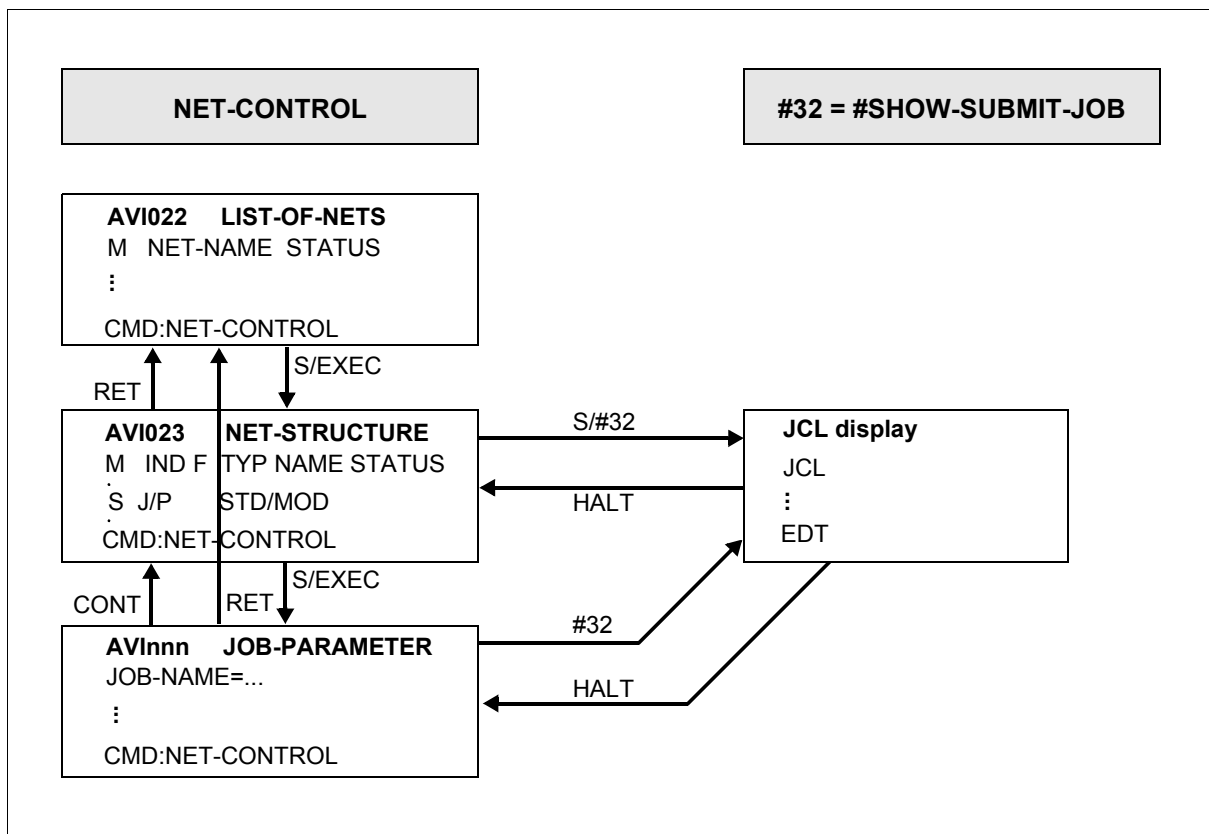


Figure 24: NET-CONTROL, CMD = #32 = #SHOW-SUBMIT-JOB

Note on AVInn

- In the case of structure elements with FU=J or FU=P mask AVI003 is displayed.
- In the case of a structure element with FU=F mask AVI026 is displayed.

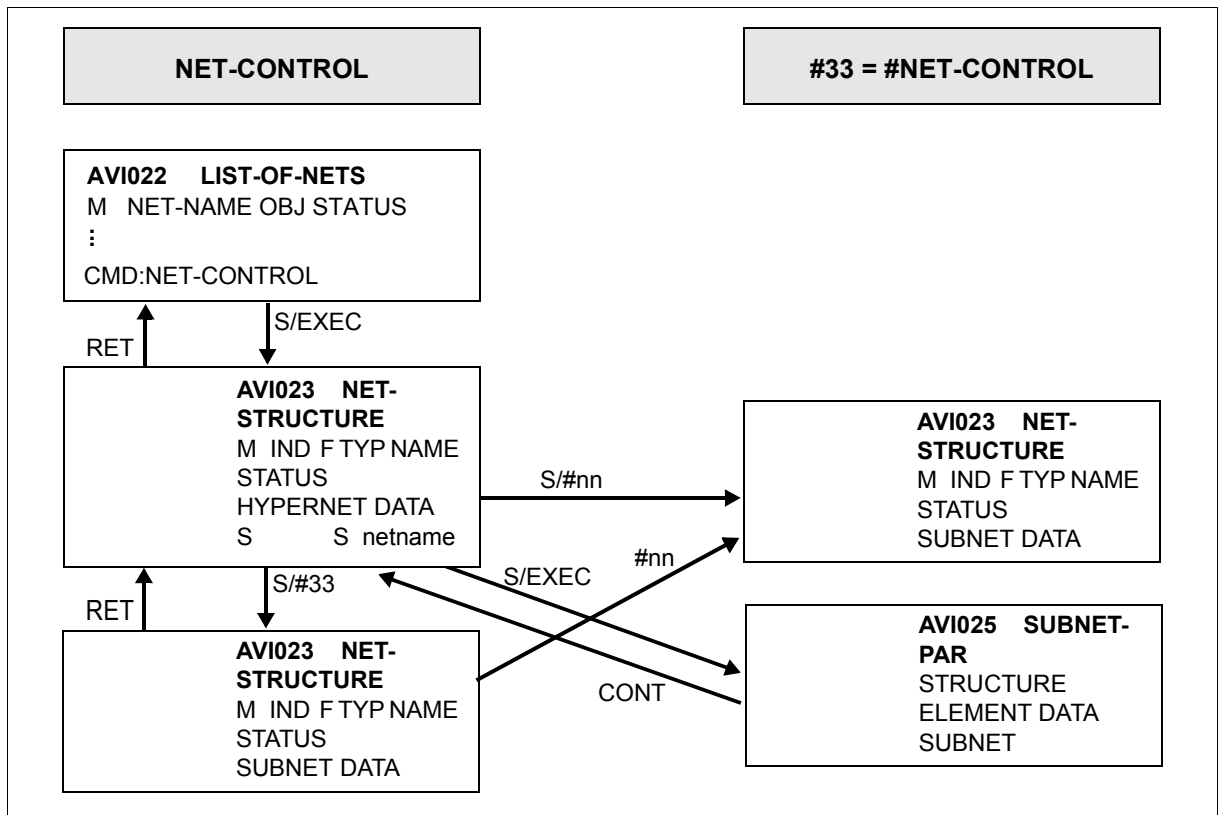


Figure 25: NET-CONTROL, CMD = #33 #NET-CONTROL (for structure elements with FU=S, Subnets)

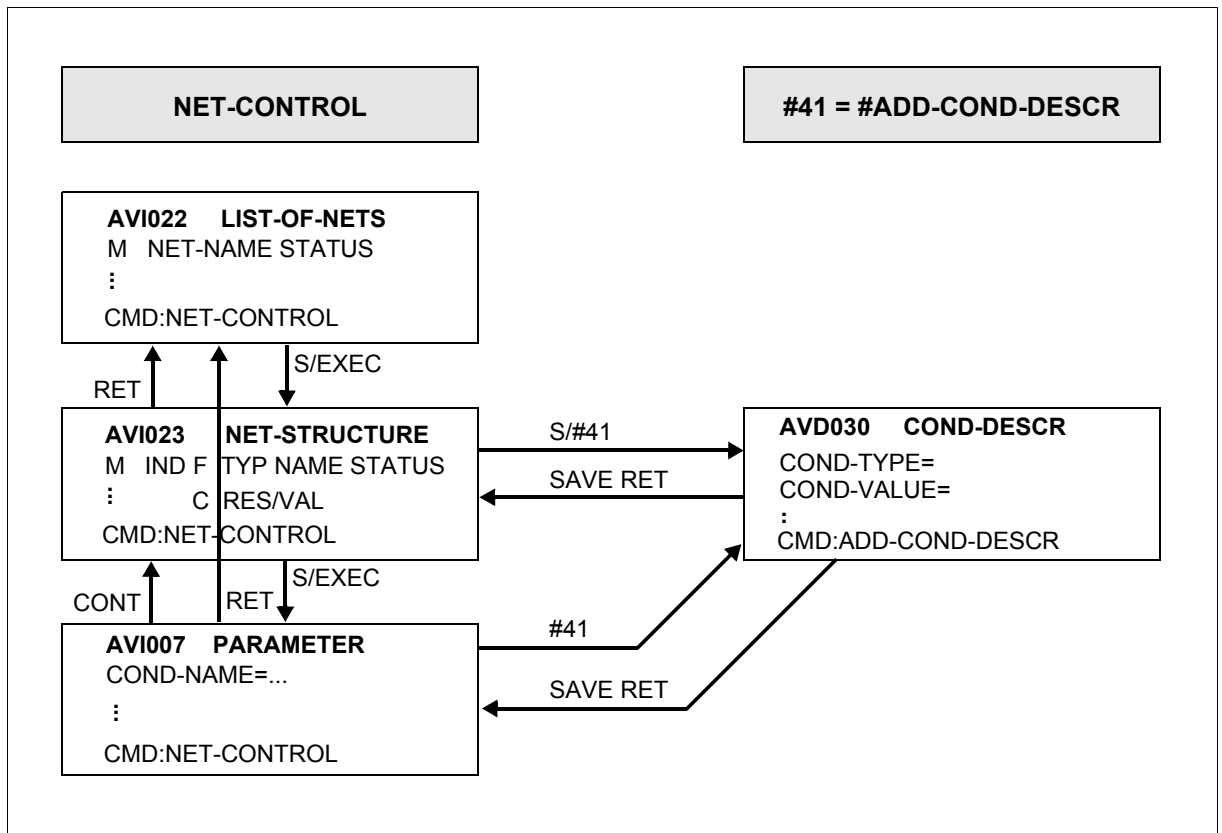


Figure 26: NET-CONTROL, CMD = #41 = #ADD-COND-DESCRIPTION

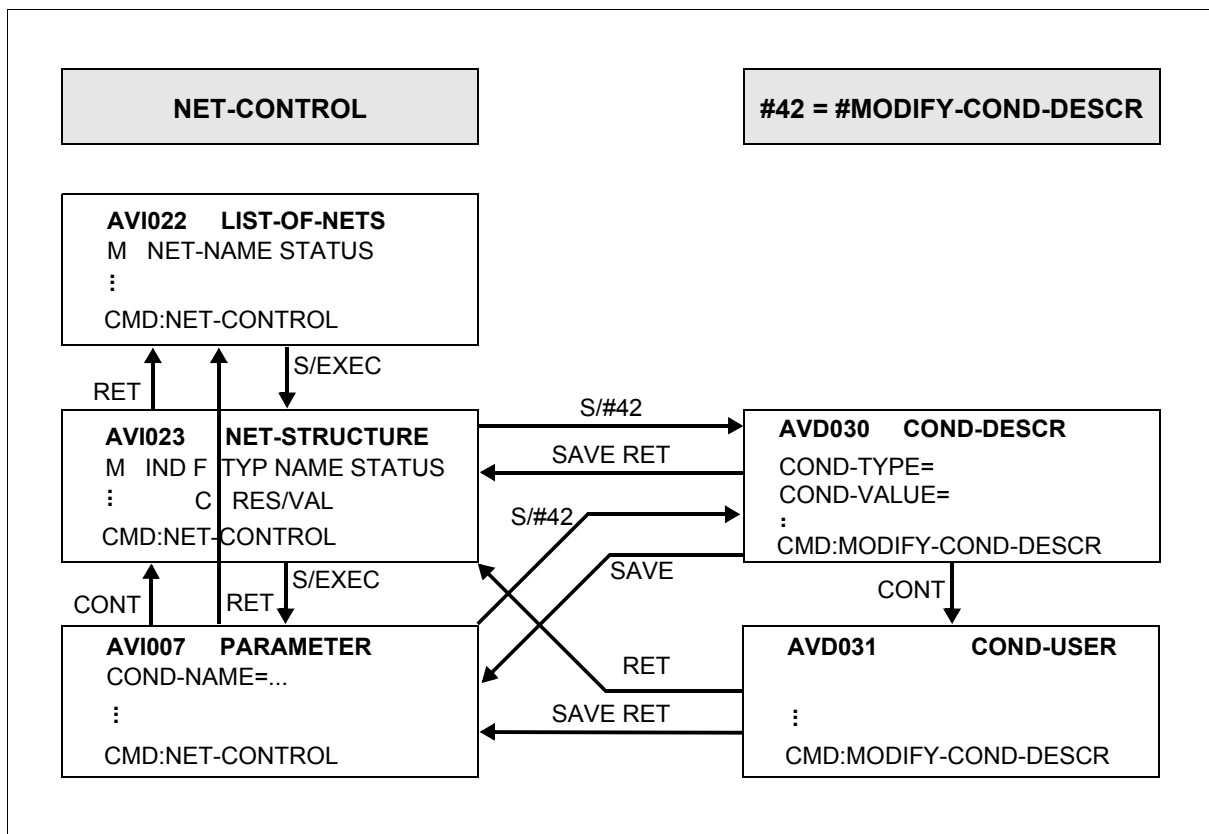


Figure 27: NET-CONTROL, CMD = #42 = #MODIFY-COND-DESCRIPTION

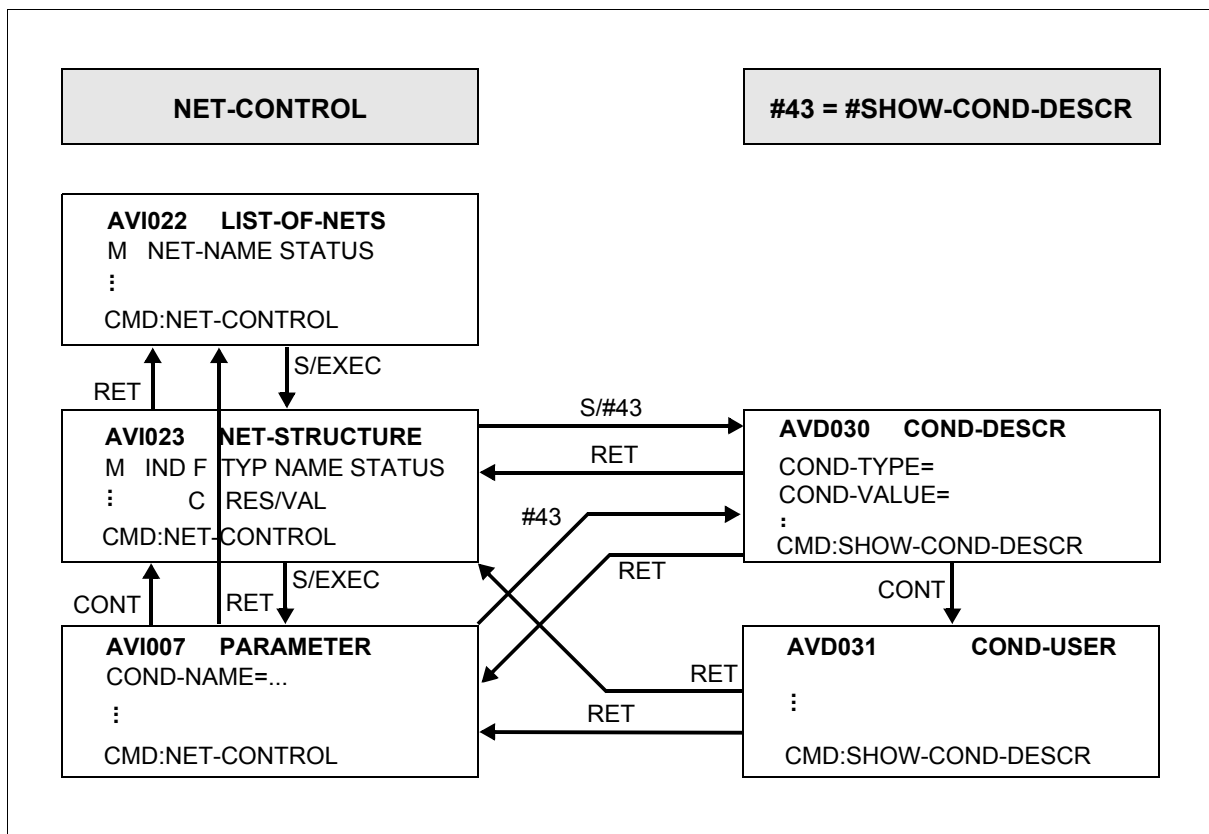


Figure 28: NET-CONTROL, CMD = #43 = #SHOW-COND-DESCRIPTION

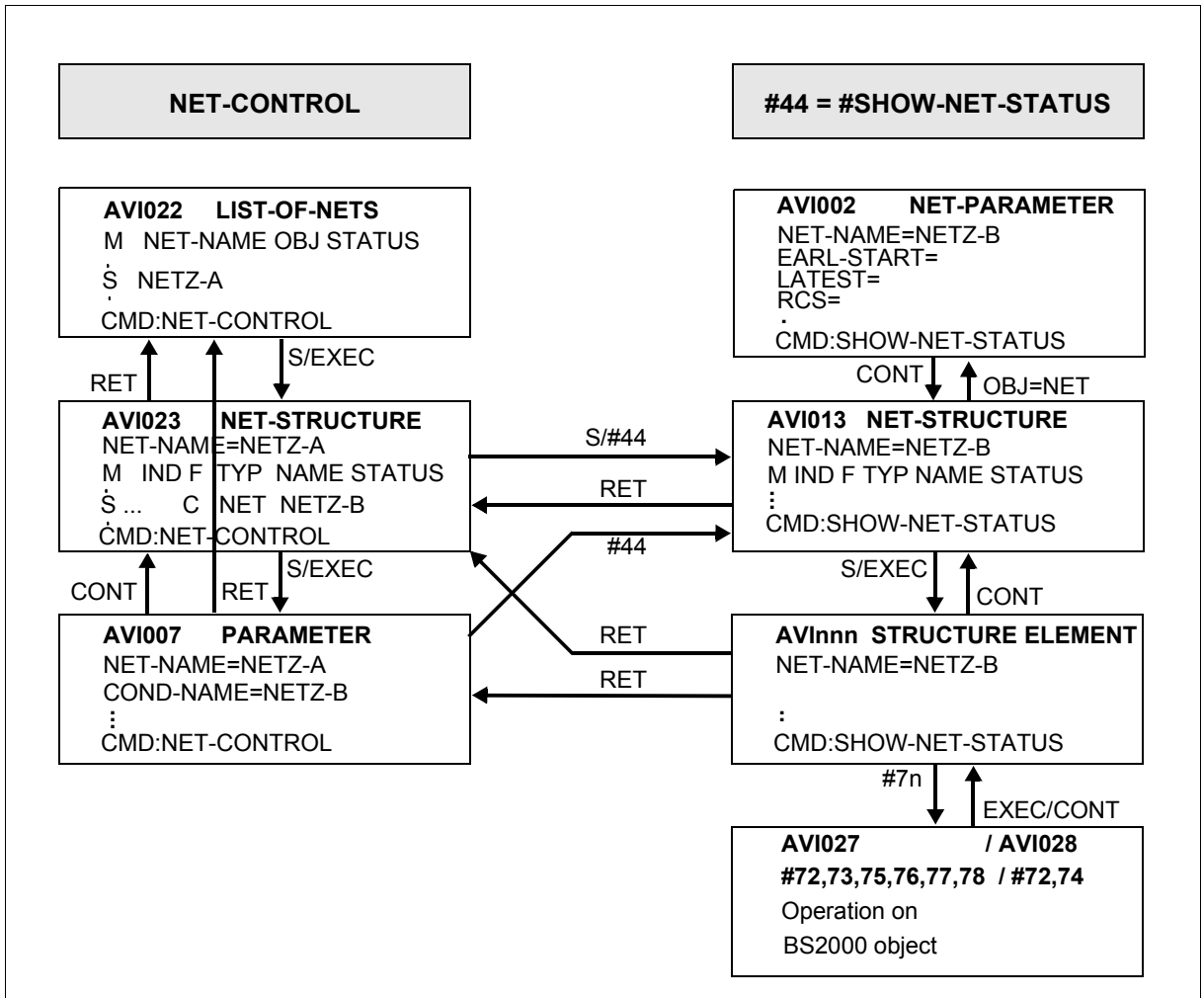


Figure 29: NET-CONTROL, CMD = #44 = #SHOW-NET-STATUS

Notes

- Depending on the structure element in mask AVI013 one of the following masks is shown:
AVI003, AVI004, AVI007, AVI008, AVI009, AVI010, AVI026
- You can branch from mask AVI003 to masks AVI027, AVI029 and AVI079 and from mask AVI004 to mask AVI028.

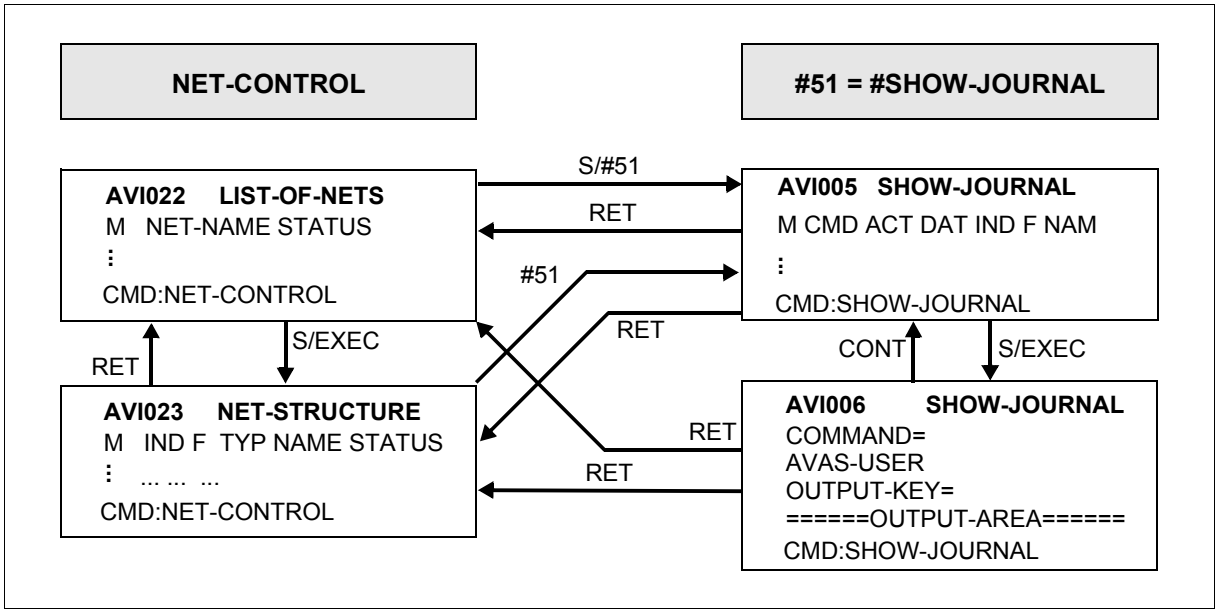


Figure 30: NET-CONTROL, CMD = #51 = #SHOW-JOURNAL

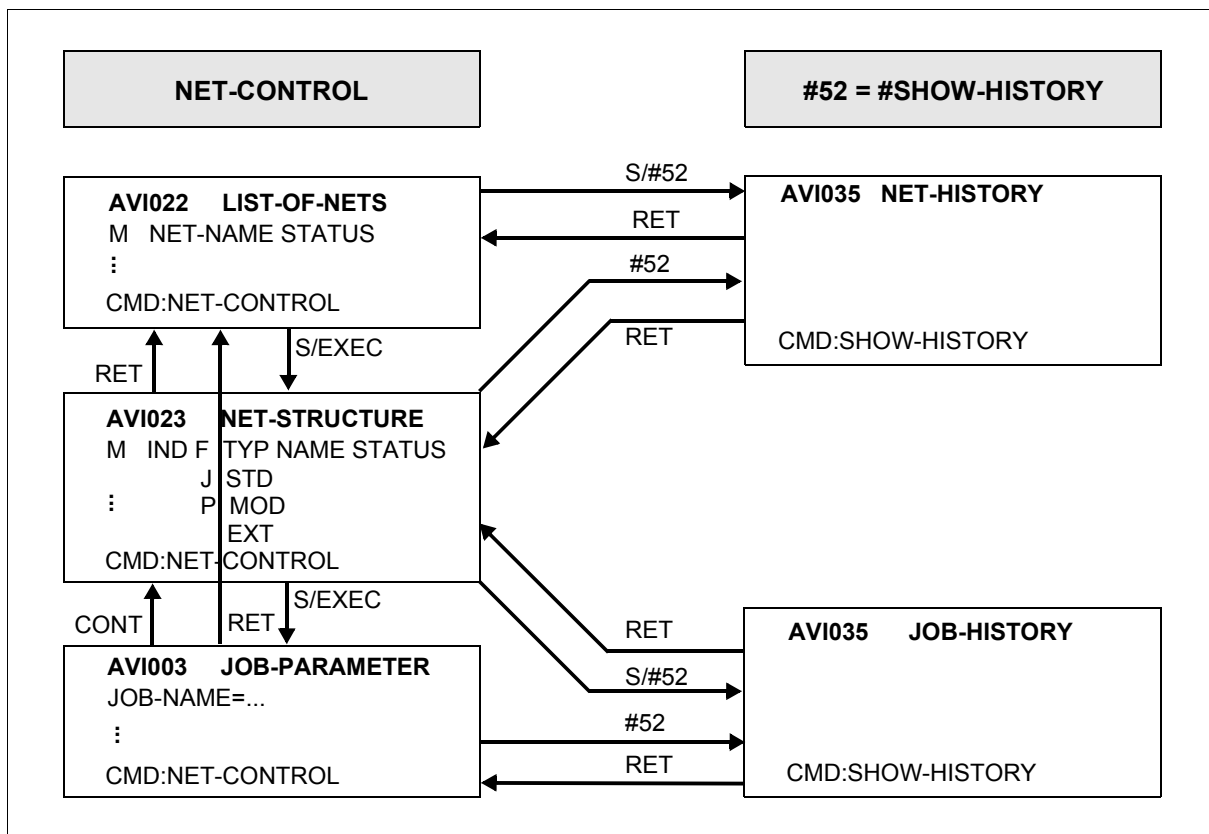


Figure 31: NET-CONTROL, CMD = #52 = #SHOW-HISTORY

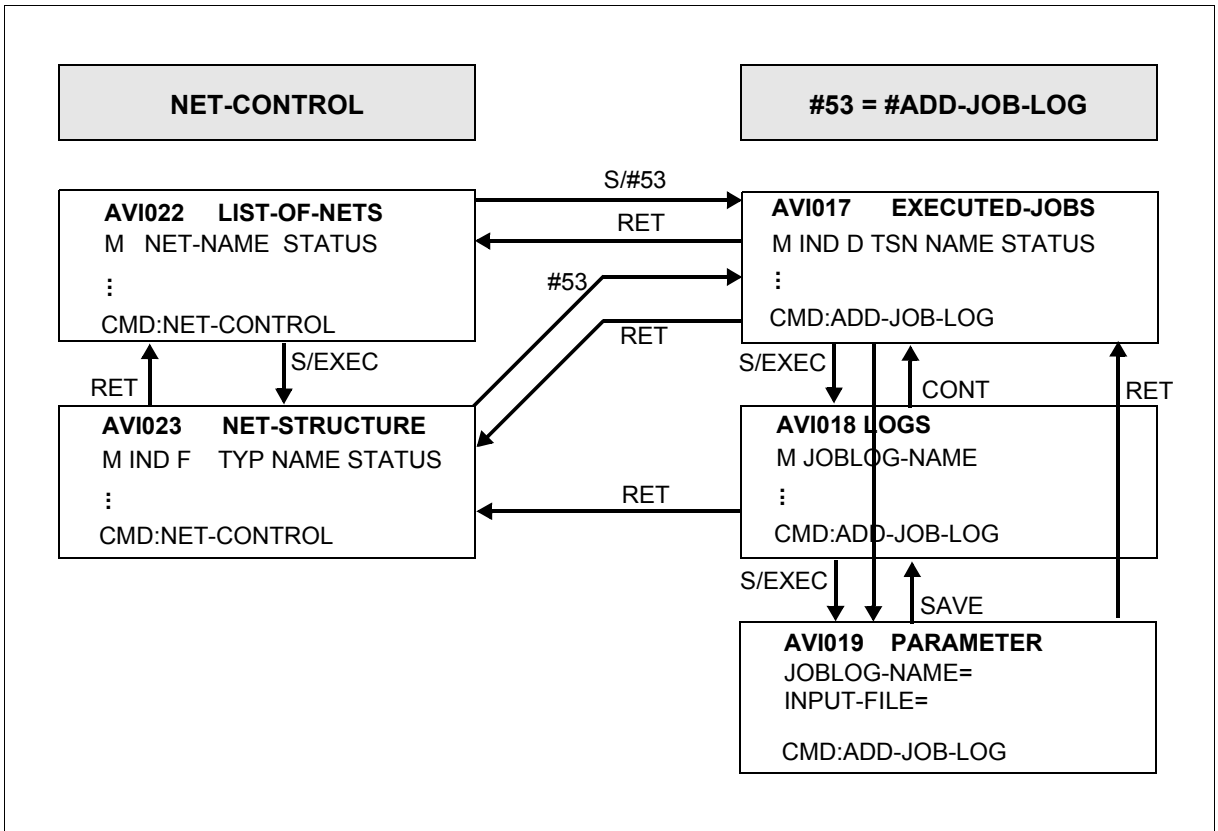


Figure 32: NET-CONTROL, CMD = #53 = #ADD-JOB-LOG

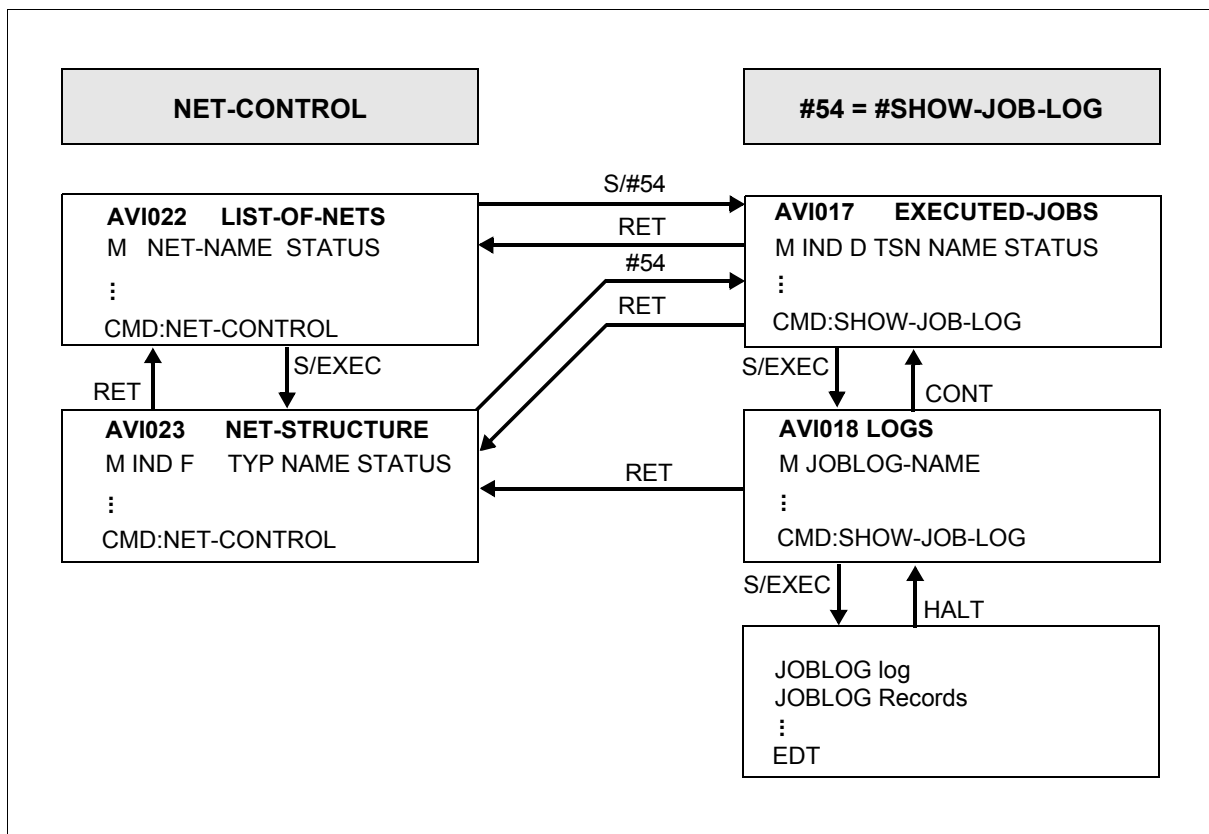
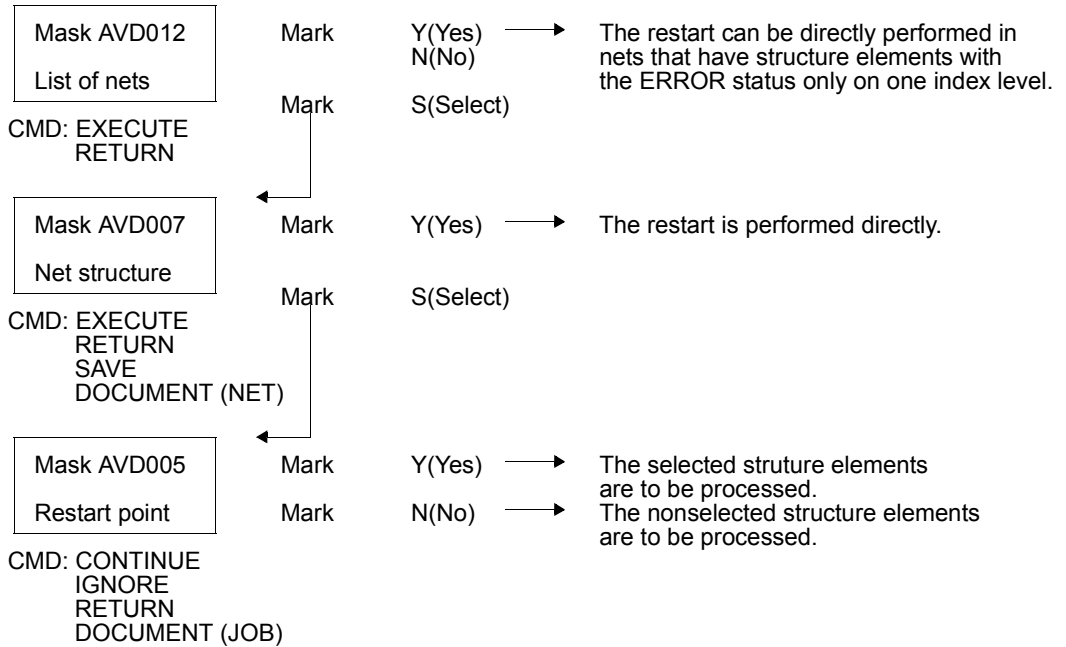


Figure 33: NET-CONTROL, CMD = #54 = #SHOW-JOB-LOG

10.5 Restart prompting



10.6 Net statuses for the statements

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
CREATE- PLAN-NET	–	TOCREATE NOTOCREATE	–	–	The net is incorporated in the production plan.
MODIFY- PLAN-NET	TOCREATE NOTTOCREATE PARTIALLY CREATED	TOCREATE NOTTOCREATE PARTIALLY CREATED	–	–	The net status is not modified.
COLLECT- NET-PARAMS	TOCREATE	TOCREATE			The net status is not modified.
CREATE- PROD-NET	TOCREATE PARTIALLY	PARTIALLY	–	–	Not all tasks in the net are produced.
	TOCREATE PARTIALLY	CREATED	–	–	All tasks in the net are produced.
MODIFY- PROD-NET	PARTIALLY CREATED	PARTIALLY	–	–	If at least one task is deleted, even if it was the last one, the status is set to PARTIALLY.
DELETE- PROD-NET	PARTIALLY CREATED	TOCREATE	–	–	All net tasks in the JMDLIB that have been modified up to now are deleted.
SUBMIT-NET	CREATED NOTTOCREATE	SUBMITTED	–	WAITING	The net is copied from the NPRLIB to the ABLDAT. Afterwards the net is in both the NPRLIB and the ABLDAT
				OPWAIT	The net waits for the START-NET statement. Afterwards the net is in both the NPRLIB and the ABLDAT.
				NETWAIT	The net waits to be started by the hypernet Afterwards the net is in both the NPRLIB and the ABLDAT.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
REPEAT-NET	-	REPEATED	-	-	A copy of a net with the SUBMITTED status is created with a new scheduled start time (NEW-PLAN-START).
				WAITING	The net waits for the start time to be reached.
				OPWAIT	The net waits for the START-NET statement.
Net start via the run control system	SUBMITTED REPEATED	SUBMITTED REPEATED	WAITING START	RUNNING	At least one task is started on the first index level.
				CONDWAIT	At least one task is started on the first index level.
				HOSTWAIT	The net waits for a host in the MSCF network. No tasks are currently executing.
				ERROR	Structure elements on the first index level cannot be processed without error.
			NETWAIT	WAITING	Structure element with FU=S/TYPE=NET is being activated. The start of the subsystem is being initiated.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
LATEST- START time passed	SUBMITTED	SUBMITTED	WAITING	WAITING	WAIT in the DELAY- SOLUTION parameter in the NET-DESCRIPTION For nets which are waiting for this net, the condition is not satisfied.
				IGNORED	IGNORE in the DELAY- SOLUTION parameter in the NET-DESCRIPTION For nets which are waiting for this net, the condition is satisfied.
				RUNNING	START in the DELAY- SOLUTION parameter in the NET-DESCRIPTION
				ABENDED	CANCEL in the DELAY- SOLUTION parameter in the NET-DESCRIPTION
HOLD-NET	SUBMITTED	SUBMITTED	RUNNING CONDWAIT HOSTWAIT RESTARTED START NETWAIT	HOLD	The net is given HOLD status if all tasks with RUNNING status have terminated.
				HOSTWAIT	At least one job in the net is waiting for a host which is currently inactive.
				HOLD	The net is not subject to control by the run control system.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
START-NET	SUBMITTED	SUBMITTED	OPWAIT WAITING	START	The net is brought to execution without taking the start time into account (only up to the next restart point of the run control system).
RESUME-NET	SUBMITTED	SUBMITTED	HOLD	RUNNING	At least one task on the following index level was started.
				CONDWAIT	The system waits at the following index level for a condition to be satisfied.
				HOSTWAIT	The net waits for a host in the MSCF network. No tasks are currently executing.
				WAITING OPWAIT RESTARTED START ERROR NETWAIT	The net is not subject to control by the run control system.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
CANCEL-NET	SUBMITTED	SUBMITTED	RUNNING	ERROR	CANCEL-TYPE=SOFT Active nets are given ERROR status when running tasks have terminated.
			CONDWAIT	CONDWAIT	CANCEL-TYPE=SOFT Inactive nets retain their net status.
			NETWAIT WAITING HOLD ERROR ENDED SHIFTED OPWAIT RESTARTED START	NETWAIT WAITING HOLD ERROR ENDED SHIFTED OPWAIT RESTARTED START	CANCEL-TYPE=SOFT The nets with the specified status are not displayed and cannot be processed.
			RUNNING CONDWAIT HOSTWAIT NETWAIT WAITING HOLD ERROR OPWAIT RESTARTED START	ABENDED	CANCEL-TYPE=HARD Nets subject to control by the run control system are given ABENDED status.
			NETWAIT	ABENDED	CANCEL-TYPE=HARD Subnets which have not been started are given the ABENDED status

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
MODIFY- SUBMIT-NET	SUBMITTED	SUBMITTED	HOLD ERROR WAITING OPWAIT START	HOLD ERROR WAITING OPWAIT START	The nets are inactive and their status is retained.
			RUNNING/ ERROR	RUNNING/ ERROR	The nets are under the control of the run control system. Only structure elements with the status ERROR may be processed. The nets retain their status.
			NETWAIT	NETWAIT	Subnets which have not been started retain their status
			NETWAIT	WAITIING	Subnets have not been started. Changing NET-TYPE > 4 to < 4 removes them from the control of the hypernet.
MODIFY- SUBMIT-JOB	SUBMITTED	SUBMITTED	HOLD ERROR WAITING OPWAIT START	HOLD ERROR WAITING OPWAIT START	The nets are inactive and their status is retained.
			RUNNING/ ERROR	RUNNING/ ERROR	The nets are under the control of the run control system. Only tasks with the status ERROR may be processed. The nets retain their status.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
Net processing via the run control system	SUBMITTED	SUBMITTED	WAITING START	RUNNING	Released net was started
				ERROR	At least one structure element of the net was processed with errors.
				ENDED	Net terminated without errors.
				HOSTWAIT	At least one structure element of the net waits for activation of a host or server.
				CONDWAIT	At least one structure element of the net waits for a condition.
			NETWAIT	WAITING	Subnet was released for execution by the hypernet.
			RUNNING	ENDED	Net terminated without errors.
				ERROR	Errors occurred during processing of at least one structure element of the net.
				HOSTWAIT	At least one structure element of the net waits for activation of a host or server.
				CONDWAIT	At least one structure element of a net waits for a condition.
			HOSTWAIT	RUNNING	Net was continued
			CONDWAIT	RUNNING	Net was continued

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
RESTART-NET	SUBMITTED	SUBMITTED	ERROR	RESTARTED	Until the next restart point of the run control system.
				RUNNING	After startup by the run control system if a task on the restarted index level was started.
				CONDWAIT	After startup by the run control system if the system is waiting for a condition to be satisfied at the restarted index level.
				HOSTWAIT	If the remote system is not available.
			RUNNING/ ERROR	RUNNING/ RESTARTED	Until the next restart point of the run control system.
				RUNNING	After startup by the run control system if the tasks at the restarted index level were restarted.
				RUNNING/ ERROR	After startup by the run control system if not all the structure elements with a status of ERROR were processed.
				ERROR	The processing of structure elements at the restart index level has again produced an ERROR status.
Reorganizing the ABLDAT	SUBMITTED	SUBMITTED	ENDED ABENDED SHIFTED IGNORED	–	The net is delete from the ABLDAT.

Statement/ function	NPRLIB		ABLDAT		Comments
	Old status	New status	Old status	New status	
DELETE- PLAN-NET	SUBMITTED REPEATED	–	–	–	The net must have been deleted from the ABLDAT (reorganizing ABLDAT). The net is deleted from the NPRLIB; it is removed from the production plan.
	TOCREATE PARTIALLY CREATED NOTTOCREATE	–	–	–	The net has not yet been released via SUBMIT-NET. The net is deleted from the NPRLIB; it is removed from the production plan.

10.7 Status of the condition descriptions for statements

COND-TYPE=NET

Statement/ Function	ABLDAT		Comments
	Old status	New Status	
SUBMIT-NET	–	CREATED	A condition description was created.
AVAK	CREATED	ENDED IGNORED ABENDED	Net processing terminated correctly. Due to the specified time having been passed, the net was not started (DELAY-SOLUTION=IGNORE). The net was not terminated correctly or was not started due to the specified time having been passed (DELAY-SOLUTION=CANCEL).
CANCEL-NET	CREATED	ABENDED	The net was terminated with the dialog function CANCEL-NET CANCEL-TYPE=HARD.
DELETE-COND- DESCRIPTION	CREATED/ ENDED/ ABENDED/ IGNORED	–	The condition description was deleted.

COND-TYPE=JOB

Statement/ Function	ABLDAT		Comments
	Old status	New Status	
SUBMIT-NET	–	NO-PLAN NO-SUBMIT CREATED	The task was excluded during planning. The task was excluded from a SUBMIT-NET. A condition description was created.
MODIFY-SUBMIT- NET	CREATED	DELETED	The task was excluded by a MODIFY-SUBMIT-NET.
AVAK	CREATED	ENDED ERROR IGNORED ABENDED	The task terminated normally. The task terminated with errors (ERROR status). The task was ignored because the specified time had passed (DELAY-SOLUTION=IGNORE). The task terminated with errors (ABENDED status).
RESTART-NET	ERROR/ ENDED/ SKIPPED/ CREATED/ ERROR	CREATED SKIPPED	The task is running again as a restart. The condition description was reset by the RESTART-NET. The task has been skipped during the RESTART-NET.
CANCEL-NET	CREATED	ERROR ABENDED	The task was abandoned due to a CANCEL-NET CANCEL-TYPE=SOFT. The task was abandoned due to a CANCEL-NET CANCEL-TYPE=HARD.
DELETE-COND- DESCRIPTION	CREATED/ ENDED/ ABENDED/ SKIPPED/ IGNORED/ NO-PLAN/ NO-SUBMIT/ DELETED	–	The condition description was deleted.

10.8 Log statuses for statements

Statement/ Function	LOGSYS		Comments
	Old status	New Status	
AVAK	–	CREATED	A log entry is created by the run control system.
SIGNAL	–	ASSIGNED	A log was signaled by SIGNAL during job execution.
TRANSFER	ASSIGNED	TRANSFERRED	The log was transferred during job execution.
	ASSIGNED	ERROR	Log could not be transferred during job execution. An error occurred.
ADD-JOB-LOG	CREATED	ADDED IGNORE	An initial log was added to this job run. No log data should be added to this log entry.
	ASSIGNED	ADDED IGNORE	A signaled log was added. Signaled log is no longer relevant.
	ERROR	ADDED IGNORE	A signaled log, which could not be transferred during job execution, was added. A signaled log is no longer relevant.
	ADDED	ADDED	A new or additional log was added.
DELETE-JOB-LOG	ADDED ASSIGNED CREATED ERROR IGNORE SAVED TRANSFERRED	–	The log or logs relating to a job run were deleted. No further processing of the log for this job run is possible.

Glossary

ABLDAT

Link name for the run control file.

ABLDUP

Link name for the copy of the run control file.

automatic restart

AVAS automatically restarts an errored net at the relevant restart point.

AVAS report

Evaluation of the AVAS production plan and the AVAS journal file according to predefined criteria.

AVAS-JV interface

Executing jobs are generally monitored using a task job variable. In the case of the AVAS-JV interface monitoring takes place using the same job variable, but this is not supplied with values as the task job variable by BS2000, but appropriate values must be supplied by the user.

AVAS-SYSTEM-LIBRARY

Name of a central AVAS library.

AVAS-USER-LIBRARY

Name of an AVAS user library.

BATCH statements

BATCH statements are selected statements which can be entered in procedures.

calendar

List of days, delimited by a start date and an end date. Each day is assigned a day of the week. Each day can also be assigned one or more symbolic start dates. Each user group is assigned a standard calendar. Nets can also be assigned to a specific calendar. Calendars are stored and managed under unique names in the CALLIB library.

CALLIB

Link name for the calendar library.

condition

Prerequisite for starting a net or an index level of a net; see also **CONDITION-TYPE**.

condition description

Part of the ABLDAT for conditions of type NET/JOB/RES/VAL; a record contains all the necessary information for the **CONDITION-TYPE** concerned.

CONDITION-JVA-NAME

Name of the job variable which has to assume a desired value at a certain position in order to satisfy the structure variable's condition.

CONDITION-TEXT

Brief description of the structure element.

CONDITION-TYPE

The type of a structure element which specifies a condition (NET/JOB/RES/VAL/JVA/TIM/TRA). Accordingly the following terms are used: Condition NET, Condition JOB (also includes conditions of the type TYP=TRA), Condition RES(OURCE), Condition VAL(UE), Condition JVA, Condition TIM

CONDITION-VALUE

Value of a condition description of a job variable.

configuration file

The configuration file is used to assign a real connection between the BS2000 system and a server system to the symbolic name of a connection to a server system (SERVER-NAME).

dependency

Situation where a net or an index level of a net is waiting for an event to occur before the start can take place.

DELAY-SOLUTION

Measures to be taken if a net is not started at the appropriate time.

DOCLIB

Link name for the library containing the documentation elements.

DOCSYS

Link name for the central library of documentation elements.

DUE key

Same as ENTER key (qv).

EARLIEST-START

Resolved start time provided for the net. It is formed with CREATE-PLAN-NET and can be modified by means of MODIFY-PLAN-NET and SUBMIT-NET. It is a search criterion when nets are selected via the operand PERIOD-NAME, but it is **not** part of a name.

ENTER-FILE

This file is used to store the JCL of jobs not managed by AVAS (jobs with JOB-TYPE=EXT).

ENTER-PARAMS

Specifies whether values should be assigned for the ENTER parameters from the net description or from the jobs.

ENTER key

Triggers transfer of the data in a mask to AVAS.

EXTERNAL-FILE

Name of an external PLAM library or SAM file as an input or output file for transferring AVAS library elements.

FILE-PASSWORD

Password for the ENTER-FILE.

FORMAT-NAME

Name of a user mask.

FT control record

Part of the net description. It describes the position of an FT request within the net as well as its parameters.

FT request

File transfer which was requested using *openFT* (TRANSFER-FILE command, see the “*openFT* User Guide” [5]).
The request is defined fully by the entries in the AVAS structure element and handled using *openFT*. Runtime monitoring and CONDITION handling takes place in the same way as for jobs.

FT-STATUS

Status indicator of an FT request.

FT-TEXT

Brief description of the FT request.

FUNCTION (also FU or F)

The function which a structure element performs within the net description. FUNCTION can take on the following values:

J (Job)	The specification required to execute a job
F (File Transfer)	The specification required to execute an FT request
P (Procedure)	The specification required to execute an S procedure
S (Start)	Description for starting a subnet
A (Add)	Create a condition description
M (Modify)	Amend a condition description
D (Delete)	Delete a condition description
C (Compare)	Test a condition descriptions
W (Wait)	Wait until a date and time

hypernet

A hypernet is a net with structural elements of type FU=S. Subnets can be run and monitored in it.

index level

Hierarchy level of the net structure. The structure elements of one index level are processed or brought to execution simultaneously. The index levels are processed consecutively in ascending order if the index level was terminated normally. If errors occur, processing is interrupted at the end of the index level involved. The sequence in which an index level is processed (or waited for) can be broken by specifying a synchronization index (SYNC-INDEX).

JCL element

Externally stored JCL of one or more jobs or S procedures. It is reincorporated in the job or S procedure via an AVAS statement within the framework of parameter modification.

JCLLIB

Link name for the library of jobs, S procedures, server jobs and JCL elements.

JCLSYS

Link name for the central library of jobs, S procedures, server jobs and JCL elements.

JMDLIB

Link name for the library of modified jobs, S procedures and server jobs.

JMDSYS

Link name for the central library of modified jobs, S procedures and server jobs.

Job

BS2000 job, FT request (without JCL), S procedure or server job

JOB

BS2000 command sequence beginning with '/SET-LOGON-PARAMETERS' and ending with '/EXIT-JOB' or '/LOGOFF'. It is also possible to incorporate special AVAS statements in the command sequence.

JOB-ACCOUNT

Parameter for the ENTER call of the job, S procedure or server representative.

JOB-CAT

Catalog ID of a SLAVE processor or server name of a remote processor.

JOB-CLASS

Parameter for the ENTER call of the job, S procedure or server representative.

job control record

Part of the net description. It describes the position of the job or S procedure within the net as well as its parameters.

JOB-DOC

Name of the documentation element for a job, an S procedure or a server job.

JOB-INDEX

Index level of a job, an S procedure or a server job in the net.

JOB-LOG

Job execution logs stored under AVAS.

JOBMAP

Link name for the library of user masks related to individual jobs or S procedures.

JOB-PARAMETER

Parameter for the ENTER call of a job, an S procedure or a server representative.

JOB-STATUS

Status indicator of a job, an S procedure or a server job.

JOB-TEXT

Brief description of the job, S procedure or server job.

JOB-TYPE

Indicates how the JCL of a task (job, S procedure) is managed in the AVAS system and how the task is monitored via a job variable (STD/MOD/EXT/ EXX).

journal file

Output medium for logging the actions of the user on the AVAS system as well as the run control system activities.

JRLDAT

Link name for the emergency journal file.

JRNDAT

Link name for the journal file.

JVA-LENGTH

Length of the value of a job variable.

JVA-NAME

Name of a job variable.

JVA-PASSWORD

Password for a job variable.

JVA-POSITION

Start position of the value in the job variable.

LATEST-START

Latest start time for the net or a task in the net.

LIFE-TIME

Time span relative to PLAN-START; indicates how long the event 'end of net' or 'end of job' is to remain valid and recognizable.

LOG

Parameter for the ENTER call of the job, S procedure or AVAS agent AVSSINCM.

LOGSYS

Link name for the central job log library (AVAS pool).

M

Column in the AVAS system masks where marks are entered to select elements.

net

Set of consecutive jobs, S procedures or server jobs whose execution is structured and defined in accordance with their logical and temporal interdependencies.

NET-ACCOUNT

Default value for JOB-ACCOUNT.

NET-CAT

Catalog identifier of a slave processor or server name of a remote processor.

NET-CLASS

Default value for JOB-CLASS.

net control record

Part of the net description. It contains parameters valid throughout the net.

NET-DELAY-SOLUTION

Action for untimely net start.

net description

Structure description of the net and information on the contents and sequence of processing steps within a net. It is created by production planning.

NET-DOC

Name of the documentation element for a net.

NETLIB

Link name for the net description library.

NET-LOG

Default value for LOG.

NETMAP

Link name for the library of user masks related to nets.

NET-NAME

Name of the net.

NET-PARAMETER

Default value for JOB-PARAMETER.

net processing

Processing of the net description (create, modify, copy, delete, display).

NET-STATUS

Status indicator for the net.

NETSYS

Link name for the central net description library.

NET-TEXT

Brief description of the net.

NET-TYPE

Control variable for serializing the processing of two or more like-named nets (but with different start times).

NET-USER

Default value for USER.

NPRLIB

Link name for the library of planned nets.

operation

Short string used to control the dialog in masks. It is entered via the CMD: field in the mask.

OUT-OF-PLAN report

This report lists nets which have exceeded a defined delay and/or which have a selected status.

PERDAT

Link name for the period file.

period

Interval delimited by start and end times. Periods are stored and managed under unique names in a separate collection of data.

PLANNED-NET-MODIFICATION report

This report lists nets which have been modified after production planning.

planning period

Time span for which selected nets are scheduled to run. It is specified via PERIOD-NAME. Those nets are processed whose symbolic start dates are entered in the calendar section corresponding to the planning period.

PLAN-START

Start time envisaged for the net during the planning operation. It is made part of the name of the nets in the NPRLIB during the planning operation and cannot be modified thereafter. The envisaged start time is modified after the planning operation using EARLIEST-START.

production plan

“Directory” for the library of planned nets, i.e. a list of the planned nets with individual resolved start times and production status.

release period

Time span during which two or more nets can be released together. It is set by the PERIOD-NAME operand. Those nets are processed whose resolved start times lie in the release period.

REPORT generator

Process for creating AVAS reports.

REPORT statements

Instructions to the REPORT generator.

resolved dependency

The event on which the start of a net or an index level depends has occurred. The condition of a waiting net has been satisfied.

resolved start time

This always consists of a date and a time of day and means that the symbolic start date of a net has been replaced by a real date. This operation takes place at the “production planning” stage.

RESTART-INDEX

Index level at which restart is to take place.

restart job

Additional job that must be performed following an interrupt before normal processing can resume.

RESTART-NAME

Name of the structure element at which any required restart is to take place.

RESTART-NET

Restart of a previously interrupted net.

restart statement

Facilities incorporated in the JCL for restart following an error.

RESTART-TYPE

This indicates whether restart statements are to be processed in a restart situation.

RESTART-VARIANT

This indicates which of the three possible restart variants is to be processed. Description of the three restart variants, consisting of RESTART-TYPE, RESTART-INDEX, RESTART-NAME.

run control file

File containing all the information needed to control execution of linked run control systems. At the "release for production" stage, the structure description of the planned net and the corresponding JCL are added to the run control file.

run control system (RCS)

This consists of an AVAS run control and monitoring routine with the name defined at generation time (RUN-CONTROL-SYSTEM), as well as all nets assigned via the run control file and the jobs brought to execution within the nets.

RUN-CONTROL-SYSTEM

Name of the run control system.

SELECT-TURNUS

Processing cycle (monthly, weekly, daily, etc.), which is always assigned a numeric value. All jobs and conditions whose SELECT-TURNUS is 0 or equal to the SELECT-TURNUS of the net control record are taken into account for processing. SELECT-TURNUS is also used as a selection criterion when defining net run variants within the framework of net planning.

SERVER-NAME

SERVER-NAME is a symbolic name for the host on which and the user ID under which a server job is to run.

standard net

Net description generated by production planning, including all job descriptions assigned to the net.

start parameter

Start parameter of a net: LATEST-START, DELAY-SOLUTION, LIFE-TIME.
Start parameter of a structure element: LATEST-START, DELAY-SOLUTION

static jobs/server jobs

Jobs/server jobs in the JMDLIB which may be assigned to two or more nets.

structure element

Individual element of a net structure for starting a task, editing a condition task or querying a condition.

subnet

A subnet is a net that is started as a structural element of a hypernet. A subnet cannot start other subnets.

symbolic start dates

Dates for the net start time, assigned when standard nets are generated and processed. They are entered in the net parameter PLAN-START. The AVAS administrator enters the symbolic start dates in the calendar and also takes charge of their further management with regard to the calendar. Symbolic start dates are also selection criteria for defining net run variants during net planning.

SYMDAT-NAME

Name of a symbolic start date.

SYNC-INDEX

Synchronization index in the net description.

task

BS2000 job or SDF-P S procedure

temporary jobs/server jobs

Jobs with the name <netname_jobname> in the JMDLIB which can be assigned uniquely to a net.

USER

Parameter for the ENTER call of the job, the S procedure or the AVAS agent AVSSINCM.

user group

Group of users who access public AVAS libraries.

USER-PARAM-FILE

User file with current values of the net run parameters supplied to the jobs of a net during production for the planned process.

Related publications

You will find the manuals on the internet at <http://manuals.ts.fujitsu.com>. You can order printed versions of manuals which are displayed with the order number.

- [1] **AVAS (BS2000)**
AVAS Statements
User Guide
- [2] **AVAS / AVAS-SV-BS2**
AVAS for the Administrator
System Administrator Guide
- [3] **HIPLEX MSCF (BS2000)**
BS2000 Processor Networks
User Guide
- [4] **SDF-P (BS2000)**
Programming in the Command Language
User Guide
- [5] **openFT for BS2000**
Enterprise File Transfer in the Open World
User Guide

Index

A

ABLDAT 209
ABLDUP 209
access control 133
 personalized 21
administration 133
assigning variable fields in masks 52
authorization concept 21, 133
authorization, privileged 134
automatic restart 209
AUTOMATIC=NO 59
AUTOMATIC=YES 58
AVAS
 master processor 119
 slave processor 119
 with HIPLEX MSCF 119
AVAS administrator 135
AVAS data stock, reading 20
AVAS exits 136
AVAS files, reorganization 137
AVAS journal file, evaluating 20
AVAS master 21, 119
AVAS pool 103
AVAS production plan, evaluating 20
AVAS reports 20, 209
AVAS slave 119
AVAS statements
 as /REMARK commands 51
 as /WRITE-TEXT commands 51
 function 51
 overview 50
 using 50
AVAS system 21
AVAS system settings 135
AVAS system variable, defining 18

AVAS variables

 overview 51
 using 50
AVAS-JV interface 209
AVAS-QUER 20
 CSV format 20
 database 20
 INFORMIX database 20
 LOAD format 20
AVAS-SV-BS2, controlling remote systems 124
AVAS-SYSTEM-LIBRARY 209
AVAS-USER-LIBRARY 209

B

batch interface 22
BATCH statements 131, 209
branch defined 62
BS2000 job 17, 48
BS2000 multiprocessor operation 21, 119
BS2000 system
 remote 125
 socket connection 125

C

calendar 18, 71, 209
 time scheduling 68
CALLIB 210
CC exit AVEX0401 89
CHANGE-NET-DESCRIPTION 34
CHECK function 34
computer network 21
COND-DOC 30

- condition 210
 - earliest time of testing 44
 - entering 30
 - external 34
 - for processing a structure element 39
 - in a net description 30
 - in a net description, defining 40
 - in status NO-OCCURE 62
 - in status OCCURRED 62
 - JOB 40
 - JVA 40, 44
 - name of 30
 - NET 40
 - NET satisfied 30
 - RESOURCE 40, 41
 - restart variant 44
 - TIM 44
 - timing control 45
 - VALUE 40, 43
- condition control 44
- condition description 41, 42, 210
 - defining 33
 - initializing in ABLDAT 40
 - lifetime 40
 - modifying 90
 - outputting contents 44
 - RESOURCE 43
- CONDITION-JVA-NAME 210
- CONDITION-NAME 30
- CONDITION-TEXT 210
- CONDITION-TYPE 210
- CONDITION-VALUE 210
- COND-STATUS
 - FREE 42
 - SHARE 42
- configuration file 210
- COPY-NET-DESCRIPTION 34
- CREATE-NET-DESCRIPTION 34
- D**
- database queries for AVAS-QUER 21
- date
 - real 18
 - symbolic 68
- defining
 - condition descriptions 33
 - structure elements for starting a job 33
 - structure elements for starting a S procedure 33
 - structure elements for starting a server job 33
- DELAY-SOLUTION 38, 94, 210
- DELETE-NET-DESCRIPTION 34
- dependency 210
 - resolved 217
- display, job runtime log 20
- DOCLIB 210
- DOCSYS 210
- documentation element 26, 28
 - assignment 54
 - COND-DOC 30
 - JOB-DOC 28
- DUE key 211
- E**
- EARLIEST-START 211
 - modification 36
- ENTER job 17
- ENTER key 211
- ENTER-FILE 28, 211
- ENTER-PARAMS 28, 211
- entry
 - for condition 30
 - for net masks 27
 - run control system 26
- executable jobs, creating 79
- executable standard jobs, creating 80
- execution parameter 29
- external conditions 34
- EXTERNAL-FILE 211
- F**
- FHS 22
- FILE-PASSWORD 28, 211
- FORMAT-NAME 211
- FT control record 211
- FT request 211
- FT-STATUS 211
- FT-TEXT 212

function authorization table 134
FUNCTION of a structure element 212
function table 21

H

HIPLEX MSCF 21
 with AVAS 119
history file 20, 102
HOSTWAIT 127
hypernet 25, 46, 94, 212
 defining 46
 processing 92
 restart 66
 run control 95

I

IFG 52
index 33
index level 212
input data
 readying 77

J

JCL 17, 19
 element 212
 modifying 90
JCLLIB 212
JCLSYS 212
JMDLIB 213
JMDSYS 213
job 213
 aborting 90
 creating 17
 creating executable 79
 creating, editing 48
 earliest start point 44
 runtime logs 103
 stored in user file 83
job control 21
job control record 213
job log 17
job modification 78
 performing 80
job name 28

job net 17, 18, 68
job runtime log
 displaying 20
JOB-ACCOUNT 213
JOB-CAT 213
JOB-CLASS 213
JOB-DOC 213
JOB-INDEX 213
JOB-LOG 213
JOBMAP 213
job-oriented production preparation 82
JOB-PARAMETER 213
JOB-STATUS 214
JOB-TEXT 214
JOB-TYPE 214
journal 98
journal file 20, 22, 98, 214
JRLDAT 214
JRNDAT 214
JVA-LENGTH 214
JVA-NAME 214
JVA-PASSWORD 30, 214
JVA-POSITION 214

L

LATEST-OCCURE 45
LATEST-START 94, 214
licensing regulations 12
LIFE-TIME 214
linking, symdats 73
LOG 214
log statuses, overview of 116
logging
 from production planning 98
 individual worksteps 98
LOGSYS 215

M

M (column) 215
MAREN 19, 87
mask-driven user interface 22
maximum wait time 89
MAX-USING-SHARE 41, 43
modification procedure 78

modifications, time of 83
modifying, released nets 99
MODIFY-NET-DESCRIPTION 34
MODIFY-SUBMIT-JOB/-NET 89
monitor 98
multiprocessor operation 21

N

name of condition 30
net 215
 creating 17
 defining 32
 production monitoring 97
 restarting 91
 running 89
 serialization 37
net control record 215
net description 7, 26, 31, 35, 215
 copying 34
 defining conditions 40
 deleting 34
 displaying 34
 modifying 34
 setting up 34
net mask
 entry for 27
net modification 78
 after production planning 106
 performing 79
 prerequisites 78
net name 26
net processing 89, 216
 aborting 90
 interrupting 90
net specifications 26
net status
 overview 198
net status, overview 107
net structure, defining 17
net type 75
net variant 71
 selection criterion 26
NET-ACCOUNT 215
NET-CAT 215

NET-CLASS 215
NET-CONTROL 95
NET-DELAY-SOLUTION 215
NET-DOC 26, 215
NETLIB 215
NET-LOG 215
NETMAP 215
NET-NAME 26, 216
NET-PARAMETER 216
NET-STATUS 216
NETSYS 216
NET-TEXT 216
NET-TYPE 75, 216
NET-USER 216
netwide production preparation 81
NO-OCCURE 42
notational conventions 11
NPRLIB 76, 216

O

OCCURE-DELAY- SOLUTION 45
operation 216
optional settings 64
OUT-OF-PLAN 105
OUT-OF-PLAN report 20, 216
overview
 of net statuses 107, 198
 of status of condition descriptions 117

P

parameter file 18, 19
partial qualification 28
partial restart 65
PERDAT 216
period 18, 70, 216
PLANNED-NET-MODIFICATION 105
PLANNED-NET-MODIFICATION report 20, 217
planning period 217
PLAN-START 217
POINT-OF-ERROR 55
POINT-OF-RESTART 56
 status changes 61
predefined system variable 18
preparation 77

- privileged authorization 134
- processing nets 89
- production control 88
- production definition 17
- production execution 8, 17, 19, 88
- production monitoring 8, 20
 - statements 97
- production plan 7, 71, 217
 - setting up 18
- production planning 18, 71
- production preparation 7, 18, 19, 77
 - job-oriented 82
 - netwide 81
- production release 19, 86
- program interface 22

- R**
- Readme file 10
- readying input data 77
- references to other publications 11
- regulations, licensing 12
- release
 - for production 8
- release for production 19
 - statements 87
- release period 217
- released jobs modify 89
- released nets modify 89
- REMARK statement 51
- remote systems, controlling 124
- reorganization of the AVAS files 137
- report
 - OUT-OF-PLAN 105
 - PLANNED-NET-MODIFICATION 105
- REPORT generator 20, 217
- REPORT statements 217
- reports 105
- resolved dependency 217
- resolved start time 217

- resource
 - allocation 41, 42
 - deleting 42
 - in AVAS 41
 - maximum shared users 41
 - multiple allocation 42
- RESOURCE, condition descriptions 43
- restart
 - automatic 209
 - in hypernets 66
 - in subnets 66
 - job 217
 - multiple 58
 - of nets after errors 91
 - partial 65
 - statement 218
 - through index level 9nn 57
- restart index 57
- restart index level 57
- restart types 57
- restart variant 19, 91
 - display 60
 - selecting 60
 - valid 56
- RESTART-INDEX 56, 57, 66, 217
- RESTART-INDEX=END 57
- restarting nets after errors 91
- RESTART-JOB-NAME 59
- RESTART-NAME 218
- RESTART-NET 218
- RESTART-SKIP-CONDITION 64
- RESTART-SKIP-ERROR 64
- RESTART-TYPE 218
- RESTART-TYPE=NORMAL 58
- RESTART-TYPE=RESTART 57
- RESTART-VARIANT 218
- RESTART-WAIT-CONDITION 65
- RESTART-WAIT-ERROR 64
- run control and monitoring routine 89
- run control file 22, 218
- run control system 19, 92, 94, 218
 - entering 26
- run control, standard nets 89
- RUN-CONTROL-SYSTEM 26, 218

- runtime logs
 - of jobs 103
- runtime variants of the net 36
- S**
- S procedure 17, 48
- scheduling over periods 70
- SDF-P 48
- selection criterion for net variants 26
- SELECT-TURNUS 26, 218
- serialization of nets 37
- server
 - cyclical check 127
 - temporarily lock 128
- server job
 - static 219
 - temporary 219
- server monitor
 - cyclical check 127
 - functions 127
- SERVER-NAME 218
- SHOW-NET-DESCRIPTION 34
- sign, symdat link 73
- SIGNAL program 104
- socket interface 125
- SOUT task 101
- specification on net 26
- standard jobs
 - creating executable 80
- standard net 25, 219
 - run control 89
- standard periods 70
- start date
 - actual 71
 - earliest 36
 - real 18
 - symbolic 18, 68
- start dates symbolic 219
- start time
 - earliest 37
 - exceeding 39
 - latest possible 37
 - resolved 217
- starting a net
 - time specifications 36
- statements
 - production monitoring 97
 - release for production 87
- static jobs 219
- static server jobs 219
- status
 - after restart 61
 - before restart 61
 - changes in POINT-OF-RESTART 61
 - changes of jobs and conditions 62
 - of condition descriptions, overview 117
- structure element 32, 35, 62, 219
 - assigning an index 33
 - conditions for processing 39
 - defining for starting a job 33
 - defining for starting a S procedure 33
 - defining for starting a server job 33
 - WAIT for TIME 44
- structure of AVAS user masks 52
- subnet 25, 46, 71, 94, 219
 - planning 75
 - processing 92
 - production preparation 75
 - release 75
 - restart 66
 - run control system 92
 - starting 49
- symbolic date 68
- symbolic start dates 219
- symdat 25
 - linking 73
 - sign 73
- SYMDAT-NAME 219
- SYNC-INDEX 219
- SYSOUT file 101
- system variable
 - predefined 18
- systems
 - controlling remote systems 124
- T**
- task 219

- task job variable
 - checking 89
- temporary job 219
- temporary server job 219
- termination information 89
- TIM condition 44
- time of modification 83
- time periods 18, 19
- time scheduling 7, 18
 - via calendar 68
 - via periods 70
- timing control for conditions 45
- TRANSFER (program) 35
- TRANSFER program 104

U

- USER 219
- user group 22, 220
- user mask
 - different 52
 - structure of 52
- USER-PARAM-FILE 48, 220

V

- valid restart variants 56
- variable fields in masks
 - assigning 52

W

- wait time
 - maximum 89
- work steps
 - logging 98
 - logging individually 98

