User Guide - English

JV V15.0

Job Variables

Valid for:

BS2000/OSD-BC V8.0
HIPLEX MSCF V6.0
SDF V4.7
SECOS V5.3

Edition December 2010

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

Contents

1 Preface

Job variables (JVs) are storage areas which enable the exchange of information both between individual jobs and between jobs and the operating system. They have a name and a value. This value may be used to control jobs and programs. For instance, the user may specify that job B is not to be started until job A has set the predefined job variable V1 to the value “START B”. The functions required for working with job variables are provided as COBOL statements or on the command or macro level.

The user can

– generate
– modify
– interrogate
– delete

job variables or instruct the operating system to set a monitoring job value accordingly when the status of a job or program changes. The user can reference the monitored job in commands by using a job variable instead of by specifying the TSN. Job variables thus provide the user with a tool enabling flexible job control. They offer a simple way of defining dependencies between jobs. Conditional job control gives the user the option of making the execution of certain actions (e.g. job initiation) dependent on changes in the values of certain predefined job variables. This is implemented by means of commands to the operating system which are executed only when, or not until, the user-defined event occurs (job variable assumes a particular value). These events can be used in the same way as user or job switches for conditional branches within command sequences and for synchronous waiting. In contrast to switches (state ON or OFF), job variables can have a number of different possible values, combinations of which can also be given in conditional expressions.

Asynchronous waiting is also possible (a command sequence is not executed until the event occurs and only then is processing interrupted).

The functions described below are available both on a single-computer system and in a BS2000 cluster.

1.1 Objectives and target groups of this manual

This manual describes possible applications of the JV (Job Variables) software product and provides information for the non-privileged and privileged user on the administration and application of job variables.
The job variable functions described are not part of the BS2000 basic configuration. For these functions, an overview of the command interface is provided. The Assembler interface (macros) are fully described for these functions. BS2000 commands that have a JV interface (MONJV function) are summarized in a table.

A knowledge of BS2000 is required, particularly of the commands. This information is contained in the “Commands” manual [1] and in the “SDF Dialog Interface” manual [2].
The privileged user must be familiar with system administration and system generation (see the manual “Introductory Guide to Systems Support” [3]).
The programming user should be familiar with Assembler. As far as other software products that utilize job variable functions are concerned, the relevant description will be found in the corresponding product manual.

1.2 Summary of contents

The manual is organized in seven chapters with the following contents:

Chapter 1 “Preface”
contains a short description of the product JV, goals, target groups and this summary of the contents of the manual, notes on the Readme file as well as the changes since the last manual for “JV”.

Chapter 2 “Management and use of job variables”
informs you about the various types of job variables and how they are named, access protection, the values of user and special job variables, the use of job variables for job and program monitoring and conditional job control and about the input from job variables.

Chapter 3 “Commands”
contains overviews of commands used to manage job variables, job monitoring, program monitoring and conditional job control.

Chapter 4 “Macros”
contains an overview of the job variable macro calls, a section with general information on macro operands and metasyntax as well as a list of descriptions of all job variable macros in alphabetical order.

Chapter 5 “Examples”
contains examples with commands and macros used to manage job variables, for job and program monitoring and for conditional job control.

Chapter 6 “Messages”
contains information on system messages and explanations of all return codes in the program level for which there is no corresponding system message.

Chapter 7 “The privileged user”
informs the privileged user about the installation of the JV product, privileges for access to the job variables of the system, job monitoring capabilities for pubset administration and about job variables as a system monitoring object with the SAT functional unit when using SECOS.

You will find a list of abbreviations and tables as well as a bibliography and index of keywords at the end of the manual.
References in the text are written using abbreviated titles. You will find the complete title of every publication in the reference section.

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/OSD

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 “Job Variables” manual for V15.0 contains the following changes with regard to the previous edition for JV V14.0:

Pubset backup with snapsets

The snapsets can be used as a logical backup of all files and job variables of a pubset.

Description of the commands

The description of the commands, the SDF syntax notation and the section treating of the command return codes were removed from this manual. For a complete description of the commands, the corresponding syntax and the command return codes, refer to the “Commands” manual [1].

New COPY-JV command: Copy job variable

For the functional description, see „New macro COPJV“. The “Commands” manual [1] contains a complete description of the command.

New macro COPJV

The COPJV macro copies the contents of a job variable (send JV) into another job variable (receive JV). If required, the protection properties of the send JV can also be copied along with the contents, apart from an existing MONJV or CJC protection. Both permanent and temporary JVs can be copied.

Extension to the CATJV macro

The new NOSTEP operand defines whether an error has occurred, if the specified JV already exists.

Extension to the STAJV macro

The new SORT operand determines the sorting of catalog entries / path names in the output.

New special job variable

The special job variable $SYSJV.TASK-CPU-USED-LONG displays the CPU time required for so-called "long-running transactions".

On an SQ server the special job variable $SYSJV.REMAINING-BS2000-RUNTIME displays the remaining runtime which is available for the BS2000 system to shut down before the real or virtual machine stops.

1.4 Notational conventions

The following typographical elements are used in this manual:

ℹ	For notes on particularly important information
⚠	This symbol designates special information that points out the possibility that data can be lost or that other serious damage may occur.
[ ]	References to other publications within the text are given in abbreviated form followed by numbers; the full titles are listed in the “References” section at the back of this manual.
`input`	Inputs and system outputs in examples are shown in typewriter font

2 Management and use of job variables

This chapter describes:

- which types of job variables exist,
- how job variables are named, stored and protected,
- how they can be used for job/program monitoring,
- how they can be used for conditional job control and
- how they can be used as an input source during the entry of commands and statements.

The syntax and operands of the commands used are described in the “Commands” manual [1].

There are two types of job variables:

- user job variables and
- special job variables.

User job variables can be stored as permanent job variables or, if the system permits it, as temporary job variables. The user has sole responsibility for creating these job variables and providing them with values. Job variables used for monitoring jobs or programs are an exception to this rule: in this case the system enters values.

Special job variables (see page 44) contain certain operating system information to which the user has read-only access or which can only be transferred to a user job variable.

Command	Macro	Function
CREATE-JV	CATJV	Generate new job variable
COPY-JV	COPJV	Copy job variable
DELETE-JV	ERAJV	Delete job variable
MODIFY-JV	SETJV	Modify contents of job variable
MODIFY-JV-ATTRIBUTES	CATJV	Modify catalog entry for job variable
MODIFY-JV-CONDITIONALLY	CSWJV	Modify job variable contents conditionally
MODIFY-MONJV	TIMJV	Set elements in the system section of a job monitoring JV
REMOVE-JV-LINK	RELJV	Delete JV-LINK entry
SET-JV-LINK	DCLJV	Define job variable link name
SHOW-JV	GETJV	Output job variable contents
SHOW-JV-ATTRIBUTES	STAJV	Output job variable attributes
SHOW-JV-LINK	LNKJV	Output JV-LINK entry

Table 1: Commands/macros for job variable management

The MODIFY-JV and SHOW-JV commands are also available to the operator (see page 237).

2.1 Storing job variables

User job variables are generated by means of the CREATE-JV command. A job variable entry is generated in the file catalog (TSOSCAT) for the job variable. The entry contains a management part (e.g. protection attributes and name of the job variables). Following the management part, 256 bytes are reserved for storing the job variable value.

Each job variable entry in the TSOSCAT file contains:

– bytes for check information

– a fixed number of bytes for the name of the job variable

– a variable number of bytes (0 - 256) for the value

2.1.1 Permanent user job variables

A permanent job variable (JV) specified in the file catalog ($TSOSCAT) remains in existence until it is deleted by means of DELETE-JV, regardless of the duration of the job which creates it. This means that a job variable can be created by one job, used by another, and deleted by a third one, although these jobs never exist at the same time.

2.1.2 Temporary user job variables

The class 2 system parameter TEMPFILE (see the manual “Introductory Guide to Systems Support” [3]) can be used by systems support to permit the use of temporary files and job variables. If, for example, TEMPFILE='#' is set, all job variables and files with names beginning with the character # are considered to be temporary, i.e. they are automatically deleted when the job which created them is completed. The default setting in the system is TEMPFILE='NO', i.e. no temporary files and job variables are permitted. The permissible values for TEMPFILE are the characters # and @. The setting TEMPFILE='#' is assumed in the examples.
Only the generating job can access temporary job variables.

2.1.3 Special job variables

Special job variables have **no** entry in the file catalog.
During “reading” of a special job variable, the appropriate data is output from system tables (e.g. date, time; for the possible items of information see page 44).

2.2 Names of user job variables

2.2.1 Permanent user job variables

Like files, job variables are identified by means of a character string comprising three parts separated by delimiters:

1. Catalog identifier (catid) identifies the pubset (PVS) in whose file catalog (TSOSCAT) the job variable has been stored.
2. User identification (userid) identifies the user entry in the user catalog (SYSSRPM) of the pubset.
3. Job variable name (jvname) uniquely identifies the job variable within the user ID.

This three-part character string is called the “path name”.
It has the following format:

```
: c a t i d : $ u s e r i d . j v n a m e
|<---------- pathname --------------->|
```

– The following characters must be used as delimiters:
  A specified catalog identifier “catid” must be enclosed in colons.
  A specified user ID “userid” must be enclosed between a dollar sign and a period.

– A path name is incomplete if at least one of the two parts “:catid:” or “$userid.” is not specified; otherwise it is complete.

– If a path name is specified incompletely, it is completed by DMS using the following default values. The default catalog identifier of the user ID is used if “catid” is missing. The user ID of the calling job is used if “userid” is missing.

– The default catalog identifier is defined in the user entry; it can be interrogated using the SHOW-USER-ATTRIBUTES command.

– Maximum lengths including delimiters:

`:catid:`	6 characters
`$userid.`	10 characters
`jvname`	41 characters

  The total length of the pathname must not exceed 54 characters. This means that the maximum length of jvname is shortened if :catid:$userid. is longer than 13 characters. With a four-character catalog ID (6 characters including delimiters) and an eight-character user ID (10 characters including delimiters), it follows that jvname cannot be more than 38 characters long. For this reason jvname should not be more than 38 characters long so as to allow for a possible change to the catalog ID or user ID.

– If only “$.” (delimiter without user ID) is specified, the system searches for the job variable under the user ID named with the class 2 system parameter DEFLUID. Default setting: DEFLUID=TSOS.

– A path name is partially qualified if “jvname” has not been specified or ends with a period; otherwise it is fully qualified.

– In jvname the same characters as for file names are permitted:

  All letters
  Digits 0 ... 9
  Special characters - , @ , # , $, .

  The job variable name must contain at least one letter.

  The TEMPFILE character cannot be the first character for a permanent JV.
  The characters “-” and “$” are not permitted as the first character.

Example

This example is designed to illustrate the meaning of the terms “partially/fully qualified” and “incomplete/complete path name” by using various path name entries..

JV name	Partially qualified	Fully qualified	Incomplete	Complete
`:V:$PM211052.JOBVAR`		x		x
`$PM211194.JV.JOB.STEUER`		x	x	
`:A:HILFSVARIABLE`		x	x	
`JOBV.VARIABLE.A`		x	x	
`:V:`	x		x	
`$PM211052.`	x		x	
`JOBV.A.`	x		x	
`JOB.MON.`	x		x	

Table 2: Examples of path name specifications

2.2.2 Wildcard syntax in path names

In the following commands and macros, wildcards may be substituted for one or more characters within a path name:

– DELETE-JV or ERAJV
– SHOW-JV-ATTRIBUTES or STAJV
– SHOW-JV-LINK or LNKJV

Wildcards can be specified in the names of temporary and permanent user job variables and in the names of special job variables. Non-privileged users may use them within the catalog identifier and the job variable name. Wildcards may not be used in the $SYSJV name section of special job variables.

Wildcards cannot replace any delimiters in the path name. In commands, the permissible use of wildcards is indicated by the suffix with-wild (n) (with-wildcards) in the data types <filename> and <partial-filename>. The specified length n=80 or n=79 only refers to the length of the entry. Using wildcards means that the entered path name, including wildcards, may be longer than 54 characters. Nevertheless, path names formed by this character string may only contain a maximum of 54 characters. The specification of wildcards in user IDs is reserved for the privileged user. See the “Commands” manual [1]for more information on specifying wildcards.

Examples

```
SHOW-JV-ATTR JV=*ALL
```
Shows all permanent job variables under the user ID, and in a sum line the number of all permanent job variables and the total length of their values.

```
SHOW-JV-ATTR JV=**ALL
```
Shows all job variables which end with the character string “ALL”. The wildcard “*” at the beginning of the character string must be specified twice.

```
SHOW-JV-ATTR JV=///
```
Shows all job variables whose names are precisely three characters long, and in a sum line the number of all job variables that are three characters long and the total length of their values.
(e.g. :catid:$userid.ABC, :catid:$userid.DEF, etc.).

```
SHOW-JV-ATTR JV=<D,M:O>*
```
Shows all job variables whose names begin either with D or with a character from the range M through O, and in a sum line the number of those job variables and the total length of their values
(e.g. :catid:$userid.DORA,:catid:$userid.DIETER, :catid:$userid.MARTHA, :catid:$userid.NORDPOL, :catid:$userid.OTTO).

`SHOW-JV-ATTR JV=-*HA`
Shows all job variables which do *not* end in the character string “HA”, and in a sum line the number of those job variables and the total length of their values
(e.g. :catid:$userid.ANTON, :catid:$userid.HANS etc.).

`SHOW-JV-ATTR $SYSJV.M*`
Shows all special job variables which start with M.
(e.g. $SYSJV.MONAT and $SYSJV.MONTH)

2.2.3 Temporary user job variables

The name of a temporary job variable begins with the character defined in the system parameter TEMPFILE (i.e. # or @), followed by a jvname up to 27 characters in length, which may also be partially qualified:
(below, # is assumed as the defined TEMPFILE character)

```
#jvname     e.g.   #JOBVARIABLE17
                   #AUFTR.STEUER.JV2
```

The special character on its own is used to address all temporary job variables of a job, e.g. in `SHOW-JV-ATTR JV=#`.

Temporary job variables are stored under the default catalog identifier and user ID of the creating job and under an internal name. Specification of a foreign user ID or catalog identifier is not permitted.

The internal name of a temporary JV is structured as follows:

S.sys.nnnn.jvname	nnnn	=	task sequence number of the creating job
	sys	=	internal number of the processor, as the TSN is not unique in the multiprocessor network with shared pubsets

The internal name part S.sys.nnnn. is specified by the user with the TEMPFILE character. The **internal** name is always output by the system.

The maximum length of “jvname” is 27 characters.

pathname:	Up to 54 characters
:catid:$userid.S.sys.nnnn.jvname	(6+10+38=54)
S.sys.nnnn.jvname	(11+27=38) (internal name part 11 characters)
#jvname	('#'+27)

In the case of status interrogations with SHOW-JV-ATTRIBUTES, the **internal** JV name is always output. The same applies to system messages containing the JV name.

Example

```
/create-jv jv=#work
/show-jv-attr jv=#
%0000000 :1OSN:$USER1.S.100.0MH4.WORK
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/del-jv jv=#,dialog-contr=*jv-change
%  JVS0469 DELETE JOB VARIABLE ':1OSN:$USER1.S.100.0MH4.WORK'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
/
```

Note

The format of internal names of job variables and files is not defined as part of the user interface and can change from one version to another without prior notice.

2.2.4 Link names for user job variables

A link name may be defined for user job variables (as for files). Assignment of a job variable to a link name is performed with the SET-JV-LINK command; the path name of the assigned job variable is stored in the JV-LINK table for the job under the specified link name. The JV-LINK table is retained until the end of the creating job.
A link name is always assigned uniquely to a job variable. Several entries may exist under different link names for any one job variable. If a link name is reassigned, the existing entry is overwritten (implicit deletion).
The link names **SMONJVJ** and **SMONJVP** should not be used. The system creates entries under these default link names in the JV-LINK table for the job during job or program monitoring (see page 64).
A valid link name consists of up to seven characters (0..9, A..Z, #,@,$); this must be prefixed by an asterisk (*) in all macros and conditional job control (CJC) commands, and for job variable substitution. In all other commands the link name is specified without an asterisk (for reasons of compatibility, however, optional prefixing of an asterisk is allowed).

In the following commands and macros (see Table 5) the creating job can use the link name to address the relevant assigned job variable (via the operand JV=*LINK or LINK-NAME). Other jobs have no access via the defined link name. Using link names enables different names to be used for job variables in programs and procedures.
The SHOW-JV-LINK command can be used to output the entries of the JV-LINK table. Note here that entered job variables need be accessible *only* on creation of the JV-LINK entry but that a job variable displayed after that time may well no longer be accessible (e.g. because it has been deleted in the interim).

The REMOVE-JV-LINK command can be used to delete one or more entries from the JV-LINK table.

Command	**Macro**	**Function**
COPY-JV	COPJV	Copy job variable
DELETE-JV	ERAJV	Delete job variable
MODIFY-JV	SETJV	Modify contents of job variable
MODIFY-JV-CONDITIONALLY	CSWJV	Check and set job variable
REMOVE-JV-LINK	RELJV	Delete JV-LINK entry
SET-JV-LINK	DCLJV	Define job variable link name
SHOW-JV	GETJV	Output job variable contents
SHOW-JV-ATTRIBUTES	STAJV	Output job variable attributes
SHOW-JV-LINK	LNKJV	Output JV-LINK entry

Table 3: Commands or macros in which a link name can be used

Link names can also be used in conditional expressions (see page 82) and in job variable replacement (see page 70).

2.3 Access administration for job variables

The owner and eventual co-owner (the terms are explained in the following section) can specify who may access the job variable and under what conditions for every job variable. This can prevent unauthorized reading of a job variable as well as unauthorized changes to a job variable.

The following standard access control protection attributes can be defined with the JV product alone:

– Access only for the owner or for all user

– General read or write protection

– General retention period

– Read and write passwords

Significantly finer settings for the access privileges are possible using the SECOS product or with its GUARDS, GUARDDEF (default protection) and GUARDCOO (co-owner protection) modules:

– Specify read and write privileges for job variables with the help of individual access profiles (guards) for every ID.

– Specify administration privileges for job variables: Co-owners of job variables can be specified with GUARDCOO.

– Specification of default values for protection attributes of job variables with the “default protection” function.

These privileges can be assigned independent of each other to any number of user IDs or user groups.

Only the job that created the temporary job variable or systems support can access a temporary job variable. No protection attributes other than the default settings for CREATE-JV can be assigned for temporary job variables.

2.3.1 Ownership of job variables

The owner and possibly other co-owners have the right to administer job variables, i.e. to create, change, delete and specify the attributes of job variables.

The owner is the user ID in whose catalog a job variable was created. This ownership cannot be changed later on.

The co-owner is the user ID TSOS by default. This co-ownership can be changed with the help of the SECOS product: User IDs can be defined as co-owners, and the co-ownership for TSOS is also limited. One exception to this are the temporary job variables. There is no way to control co-ownership for these job variables.

Some protection attributes are specified by the system administration (regardless of the ownership):

– The system parameter FSHARING specifies if job variables are also accessible for user IDs that do not have a user entry for the pubset of the job variable.

– When using SECOS systems support can define global pubset default values for protection attributes of job variables. These values take effect when no protection attributes were assigned when the job variable was set up and there are no user specific default values.

Controlling co-ownership

Co-ownership allows a user logged in under a user ID other than that of the owner to create, modify and delete job variables with the same privileges as the owner of the job variable.

With the GUARDCOO subsystem it is possible to define co-owners of job variables. Co-ownership allows a user logged in under a user ID other than that of the owner to create, modify and delete job variables with the same privileges as the owner of the job variable. If the GUARDCOO subsystem is not activated or an error occurs while checking the access privileges, then the co-ownership is ignored, i.e. only the common access privileges are in effect.
GUARDCOO is part of the SECOS software product.

The definition of the co-owner is done with GUARDS in two steps:

– Job variables for which the co-owners are to be specified are selected using name patterns (rules) in rule containers (guards). The name patterns are evaluated for existing job variables as well as for new job variables, but not for temporary job variables.

– You then specify who is allowed to execute which administration functions under what conditions for each guard. Co-ownership can be assigned for certain user IDs, members of certain user groups or owners of certain global system privileges.

You will find more detailed information on controlling co-ownership with SECOS in the SECOS manuals (see the “SECOS” manual [10]) under the keyword “co-owner protection”).

Restricted TSOS co-ownership

The owner can restrict the access and administration privileges of the TSOS user ID. Under the user ID TSOS the protection attributes of an external job variable (i.e. TSOS is not the owner) cannot be changed anymore, and it is also not possible anymore to delete a job variable when bypassing the protection attributes (IGNORE=ACCESS): If TSOS co-ownership is set to *RESTRICTED for a name pattern, then the IGNORE= ACCESS specification is ignored and the privilege check based on the protection attributes for ACCESS, basic ACL and GUARDS is used instead.

See also the GUARDS command /ADD-COOWNER-PROTECTION-RULE in the “SECOS” [10] manual. Primarily affected by the restriction are the JV commands /DELETE-JV and /MODIFY-JV-ATTRIBUTES (see “Commands” manual [1]).

The restriction of co-ownership for job variables has no effect under the TSOS user ID because TSOS is the (permanent) owner of the job variables.

You will find more detailed information on the restriction of co-ownership with SECOS in the SECOS manuals (see the “SECOS” manual [10]) under the keyword “co-owner protection”).

2.3.2 Protection mechanisms for access control

The following access attributes can be controlled for a job variable (the name(s) of the protection attribute(s) is (are) shown in the parentheses):

- Users or user groups with access (shareable, multi-user access)
- Read and/or write authorization, possibly separated according to access privileges (type of access, multi-user access)
- Passwords for access (passwords)
- Limitation (with respect to time) of the access (retention period)

Protection mechanism	Protection capabilities	Granularity of access privileges
GUARDS ↑	– Further differentiation of multi-user access and type of access	Individual access profiles ↑
Basic ACL ↑	– Differentiation of shareable and type of access	Access profiles for groups ↑
Default protection	Default access control – Type of access (read/write access) – shareable Additional protection capabilities – Passwords – Retention periods	Owner or all

Table 4: Overview of protection capabilities for a job variable

The protection capabilities for a job variable are arranged on the three default protection levels (lowest level), basic ACL and GUARDS (highest level) (see table 4). The strongest active protection mechanism always applies for the protection of a job variable.

The use of passwords and the retention period (release date) is independent of the active protection mechanism, i.e. it is also possible for basic ACL and GUARDS. Access control (shareability and type of access) are further refined for these two protection mechanisms by creating three groups (OWNER, GROUP and OTHERS) for basic ACL and by creating access profiles for individual users for GUARDS.

Coexistence of the access mechanisms

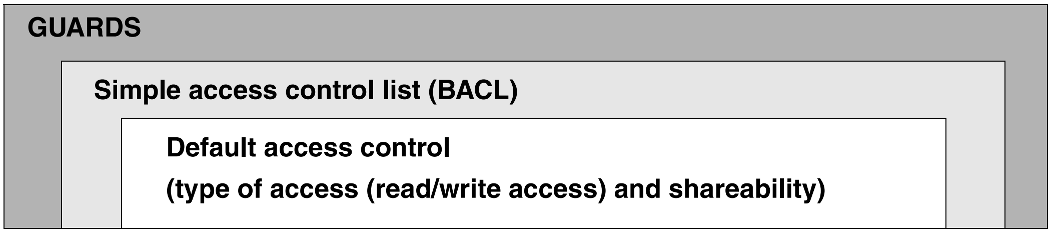

GUARDS
Simple access control list (BACL)
Default access control
(type of access (read/write access) and shareability)

Figure 1: Hierarchy of protection mechanisms for job variables

figure 1 clearly shows that the higher protection levels completely include the functionality of the levels below them.

The following rules apply where standard access control (USER-ACCESS, ACCESS), basic ACL as well as guards are used:

1. The values of the USER-ACCESS or ACCESS operand are entered in the catalog regardless of the fact that a higher-level protection mechanism (basic ACL or guards) may currently be in effect. If this is the case, the entries will not be evaluated until the higher-level protection mechanism is deactivated.
2. The BASIC-ACL operand sets the BASIC-ACL indicators in the catalog entry for the job variable, regardless of whether the job variable is protected by basic ACL or a higher-level protection mechanism (guards). If a higher-level protection mechanism is active, the entries will not be evaluated until the higher-level protection mechanism is deactivated.

Default access protection

The default access control mechanism provides the protection attributes type of access and shareable. In addition, a retention period and password protection can be specified for a job variable in the default access protection (but also for basic ACL and GUARDS).

Type of access

A user can specify which type of access to a job variable he wants to permit using the operands of the CREATE-JV and MODIFY-JV-ATTRIBUTES commands.

ACCESS=*BY-PROTECTION-ATTR (default value)	The type of access is irrespective of the value of the operand PROTECTION-ATTR (see section “Default protection (user defined default values)” on page 36)
ACCESS=WRITE	Read and write access is permitted
ACCESS=READ	Read access is permitted

At the macro level (CATJV macro) the operand value PROTECT=DEFAULT corresponds to the operand value ACCESS=*BY-PROTECTION-ATTR for the commands. The PROTECT operand refers not only to the type of access, but to all protection attributes. The operand values for read and write access are identical to those of the command operands.

Shareable

In the CREATE-JV and MODIFY-JV-ATTRIBUTES commands you specify which user IDs can access the job variable with the previously assigned access types with the USER-ACCESS operand (shareability):

USER-ACCESS=*BY-PROTECTION-ATTR (default value)	The access irrespective of the value of the operand PROTECTION-ATTR (see section “Default protection (user defined default values)” on page 36)
USER-ACCESS=*OWNER-ONLY	Only the JV owner has access
USER-ACCESS=*ALL-USERS	All user IDs have access

At the macro level (CATJV macro) the operand values SHARE=NO / YES correspond to the values USER-ACCESS=*OWNER-ONLY/ *ALL-USERS.

Retention (protection) period

When a retention period is specified a job variable can be protected against changes and deletion for a specified period of time.
When the job variable is created, the current date is entered in the EXPIR-DATE output field of the SHOW-JV-ATTRIBUTES command by default. This date is then updated automatically when the contents of the job variable are changed.

You can specify for how many days from the current date the job variable will be protected from changes and deletion in the RETENTION-PERIOD operand using the command MODIFY-JV-ATTRIBUTES. When a retention period is specified, the number of days specified is added to the current date. The date (in EXPIR-DATE) now shows when the job variable will be able to be changed again.
The entry in the EXPIR-DATE output field remains unchanged until the retention period expires. After that, it is automatically modified to reflect the current date when the contents of the job variable are changed.

At the macro level (CATJV macro) the operand RETPD correspond to the operand RETENTION-PERIOD.

Password protection

Furthermore, you can make access to a job variable dependent upon the knowledge of a password.
A read password can be defined in the CREATE-JV or MODIFY-JV-ATTRIBUTES command in the operand READ-PASSWORD and a write password can be defined in the WRITE-PASSWORD operand.

The value YES in the job variable entry in the READ-PASS and WRITE-PASS fields shows that a read or write password has been defined. If no password is defined, then the value is NONE.
A new job variable is not protected by a password by default.
The required password must be entered in the password table of the entry with the ADD-PASSWORD command before accessing the job variable. This entry is valid until the job is completed. In some commands the password is specified in the PASSWORD or JV-PASSWORD operand. Access is only authorized then when the command is executed.
If a password is defined, then this password must be specified by all users who do not have system administration privileges, even when the protection attributes are to be changed with the MODIFY-JV-ATTRIBUTES command.

Password protection		**Additional protection using a password**	**Required specification for**	
READ-PASS	**WRITE-PASS**		**Read access**	**Write access**
NONE	NONE	No protection	None	None
YES	NONE	Read access Write access	Read password	Read password
NONE	YES	Write access	None	Write password
YES	YES	Read access Write access	Read or write password	Write password

Table 5: Password protection combinations and specifications for read or write access

When an attempt is made to access a job variable without the required password, then the non-privileged user is penalized with a time penalty. The duration of the time penalty depends on the setting for the system parameter PWPENTI and can have a value from 0 to 60 seconds. The user cannot input data during this time.

In the SHOW-JV, MODIFY-JV, MODIFY-JV-CONDITIONALLY, MODIFY-MONJV, DELETE-JV commands and the corresponding macros the password can still be specified in the command (PASSWORD operand). When the wrong password is entered, a check is made to see if the password has already been entered in the password table. If the password is not in the table, then the access attempt is an unauthorized attempt.
If this is not the case, then the access is authorized and the command is executed.

At the macro level (CATJV macro) the operand values RDPASS and WRPASS correspond to the READ-PASSWORD and WRITE-PASSWORD operands.

Interface overview

Command/Macro	**Function**
CREATE-JV/CATJV	Create the job variable and specify protection attributes with the command operands ACCESS,USERACCESS, READ-PASSWORD and WRITE-PASSWORD or with the macro operands ACCESS, SHARE, RDPASS and WRPASS.
MODIFY-JV-ATTRIBUTES / CATJV	Change the protection attributes with the command operands ACCESS,USER-ACCESS, READ-PASSWORD, WRITE-PASSWORD and RETENTION-PERIOD or with the macro operands ACCESS, SHARE, RDPASS, WRPASS and RETPD.
SHOW-JV-ATTRIBUTES / STAJV	Output the protection attributes The job variables can also be selected according to specific protection attributes using the command operands ACCESS, USER-ACCESS, PASSWORD and EXPIRATION-DATE within SELECT=*BY-ATTRIBUTES(...). On the macro level, selection is possible in the STAJV macro, via the macro JVSEL with the operands ACCESS, SHARE, PASS and EXDATE.

Table 6: Commands and macros for specifying default protection attributes

The table presents an overview of the commands and macros used to specify the type of access (ACCESS), shareable (USER-ACCESS / SHARE), retention period (RETENTION-PERIOD / RETPD) and password protection (READ-PASSWORD and WRITE-PASSWORD / RDPASS and WRPASS) default protection attributes.

Example

```
/create-jv jv=status,prot=*par(access=*write,user-access=*all-users,
           read-pass=c'rdpw',write-pass=c'wrpw')
```

The job variable status is created with the following properties:

– REad and write access is permitted
– All user IDs have access
– A read protection password ('rdpw') is set
– A write protection password ('wrpw') is set

Basic ACL protection

In a basic ACL (simple access control list) the read and write access privileges can be assigned explicitly for the user classes OWNER, GROUP and OTHERS.
This protection capability does not exist for temporary job variables.
When basic ACL protection is activated, the required access privilege must be set explicitly in order to obtain access. In contrast to the default access control (with ACCESS and USER-ACCESS) in a basic ACL, write access does *not* implicitly imply that the user has read privileges. Read access must be set explicitly to have read privileges.

The users are divided into the following user classes:

OWNER is the user ID of the owner or systems support.

GROUP is a collection of all user IDs that belong to the user group of the owner. In the basic function (without SECOS) these IDs are all other user IDs since only one user group exists. When the SECOS software product is used, several user groups can be defined using the SRPM function unit. Access by users that do not explicitly belong to a different group from the owner are still evaluated according to the access privileges of the GROUP group (the owner of a job variable can only belong to a maximum of one group). Access by the users that explicitly belong to a different user group are then evaluated according to the access privileges of the OTHERS group.

OTHERS are all user IDs that explicitly belong to a different user group from the owner or that do not belong to any user group. If SECOS is not used, the access privileges are to be assigned as for GROUP to avoid changes later on when SECOS is used.

The basic ACL protection attributes are only set when at least one access privilege was assigned explicitly (basic ACL is activated). The basic ACL protection attributes are shown in the OWNER, GROUP and OTHERS output fields using the SHOW-JV-ATTRIBUTES command. The values “R W (read and write), “R -” (read only), “- W” (write only) or “- -” (no access) are shown for each user class. The fields are only shown when basic ACL is activated.

The basic ACL protection can be activated, access privileges can be changed or the basic ACL protection can be deactivated with the CREATE-JV or MODIFY-JV-ATTRIBUTES command in the basic ACL operand by explicitly setting the access privileges.
A job variable is created without basic ACL by default.

When basic ACL is activated, access control is performed according to the access privileges settings. The shareable and type of access standard protection attributes are not evaluated in this case.

When SECOS is used, several user groups can exist. This means you can also assign different access privileges to the group of the owner and other user groups.

When the basic ACL protection is activated for an existing job variable, the user can create a basic ACL by specifying basic ACL=*PREVIOUS that matches the values of the default access control (see the command MODIFY-JV-ATTRIBUTES in the “Commands” manual [1]). When a job variable is created, a basic ACL can be created by specifying basic ACL=*STD in which only the owner has all access privileges.

The macros and their operands used to specify the basic ACL protection are listed in the following interface overview together with the commands.

Interface overview

Command/Macro	**Function**
CREATE-JV/CATJV	Create a job variable and specify protection attributes with the command operands BASIC-ACL (suboperands OWNER,GROUP and OTHERS), READ-PASSWORD and WRITE-PASSWORD or with the macro operands OWNERAR, GROUPAR, OTHERAR, RDPASS and WRPASS.
MODIFY-JV-ATTRIBUTES / CATJV	Change the protection attributes with the command operands BASIC-ACL (suboperands OWNER, GROUP and OTHERS), READ-PASSWORD, WRITE-PASSWORD and RETENTION-PERIOD or with the macro operands OWNERAR, GROUPAR, OTHERAR, RDPASS, WRPASS and RETPD.
SHOW-JV-ATTRIBUTES / STAJV	Output the protection attributes The job variables can also be selected according to specific protection attributes using the command operands BASIC-ACL (sub-operand OWNER, GROUP and OTHERS), PASSWORD and EXPIRATION-DATE within SELECT=*BY-ATTRIBUTES(...). On the macro level, selection is possible in the STAJV macro via the macro JVSEL with the operands BASACL, OWNERAR, GROUPAR, OTHERAR, PASS and EXDATE.
ADD-USER-GROUP (SECOS-command)	Enter user group in the user catalog of a pubset and assign user IDs to a user group
REMOVE-USER-GROUP (SECOS-command)	Delete the user group
SHOW-USER-GROUP (SECOS-command)	Show the user group entry

Table 7: Commands and macros for specifying access protection with basic ACL protection

Example

```
/create-jv jv=jv.develop ─────────────────────────────────────────────  (1)
/add-user-group group-id=develop1,add-group-member=user1 ─────────────  (2)
/mod-jv-attr jv=jv.develop,protection=(basic-acl=(
                     owner=(read=*yes,write=*yes),
                     group=(read=*yes)
                     others=(read=*no) ─────────────────────────────────  (3)
```

(1) A job variable “jv.develop” is created

(2) A group named “develop1” is created by the owner of the job variable “jv.develop” and his own user ID (USER1 here) is entered as a member of the group.

(3) Access protection is defined using a basic ACL for the job variable “jv.develop”. The owner of the job variable has read and write access to the job variable. The members of the group to which the owner of the job variable belongs (“develop1”) may only read the job variable. All other user IDs (“others”) may not access the job variable.

GUARDS protection

Access control for job variables can be done using guards (**G**enerally **U**sable **A**ccess Cont**R**ol A**D**ministration **S**ystem). This type of access protection only takes effect when the subsystem GUARDS is loaded (part of the SECOS software product, see [10]). There is no protection capability for temporary job variables.

Access to a job variable is controlled through a special access profile (guard) that contains all condition under which access is permitted or rejected (date, time, time period, user ID). Every access profile is created with the corresponding GUARDS commands and is stored as an entry in the Guards catalog under a name assigned by the user. Every pubset has a guard catalog that is managed separately from the user files.

Access to a job variable protected by a guard is only possible when the conditions specified in the guard entry permit it.

Activating guard protection

Guard protection is activated when a value not equal to *NONE is specified in the GUARDS operand for a CREATE-JV call.

If guards are not assigned for all access privileges, then the access privileges not specified are also entered in the file catalog as “not specified” (*NONE). The access control will reject a corresponding access because no privileges were specified. Write privilege does not implicitly imply read privileges for GUARDS.

The macros and operands used to specify the GUARDS protection are listed in the following interface overview together with the commands.

Interface overview

Command/Macro	**Function**
CREATE-JV/CATJV	– Create a job variable and activate GUARDS protection with the READ and WRITE suboperands of the GUARDS operand (same operand names as for the CATJV macro) – Specification of additional protection attributes with the command operands READ-PASSWORD and WRITE-PASSWORD or with the macro operands RDPASS and WRPASS
MODIFY-JV-ATTRIBUTES / CATJV	– Activate/deactivate GUARDS protection with the READ and WRITE suboperands of the GUARDS operand (same operand names as for the CATJV macro) – Specification of additional protection attributes with the command operands READ-PASSWORD, WRITE-PASSWORD and RETENTION-PERIOD or with the macro operands RDPASS, WRPASS and RETPD
SHOW-JV-ATTRIBUTES / STAJV	Output the protection attributes The job variables can also be selected according to specific protection attributes using the command operands GUARDS (sub-operands READ and WRITE), PASSWORD and EXPIRATION-DATE within SELECT=*BY-ATTRIBUTES(...). On the macro level, selection is possible in the STAJV macro via the macro JVSEL with the operands GUARDS (suboperands READ and WRITE), PASS and EXDATE.
CREATE-GUARD (SECOS-command)	Create a guard (does not contain any protection mechanism yet)
DELETE-GUARD (SECOS-command)	Delete a guard

Table 8: Commands and macros used to specify access protection GUARDS (part 1 of 2)

Command/Macro	Function
ADD-ACCESS-CONDITIONS (SECOS-command)	Specify the access conditions of a guard
MODIFY-ACCESS-CONDITIONS (SECOS-command)	Change the access conditions of a guard
SHOW-ACCESS-CONDITIONS (SECOS-command)	Output the access conditions of a guard
SHOW-ACCESS-ADMISSION (SECOS-command)	Output the access conditions of a guard that are valid for the caller's user ID.

Table 8: Commands and macros used to specify access protection GUARDS (part 2 of 2)

A job variable is linked to a guard entry by entering a guard name in the corresponding operand of the macro/command (CATJV/CREATE-JV, MODIFY-JV-ATTRIBUTES). The link is maintained in the catalog entry of the job variable.

Example

```
/create-guard guard-name=protjv ----------------------------------------  (1)
/create-jv guardjv,protection=(guards=(read=protjv,write=*none)) -------  (2)
/add-access-conditions guard-name=protjv,
                       subjects=*user(user-identification=mueller),
                       admission=*parameters(time=*interval(from=07:00,
                                                   to=17:00)) -----  (3)
```

(1) The guard “protjv” is created.

(2) The job variable “guardjv” is created and access protection via GUARDS is activated through the link to the guard “protjv”. Read protection is controlled through this guard, write access is not permitted.

(3) It is entered in the guard “protjv” that the user “mueller” may access the job variable “guardjv” between 7:00 and 17:00.
(If the specified guard does not exist yet, then it is created automatically).

The catalog entry for the guard protection can also be created without the GUARDS subsystem. However, there are no access privileges for the job variable in this case.

If guard protection is activated for a job variable, then all access types of the GUARD operand (READ and/or WRITE) that are not explicitly set are set to *NONE. Access via these access types is **not** possible then. For example, when `/modify-jv-attributes jvtest,protection=(guards=(read=protjv))` is executed, `protection=(guards=(read=protjv,write=*none))` is set automatically.

Only when a job variable protected by guards is accessed is the check to see if the specified guard entry (guard name) exists, if it can be used and if the user is permitted to access the job variable using the corresponding type of access based on this access profile.

Note

A job variable cannot be accessed when protection via guards is entered in the catalog entry for the job variable but there is still no access profile defined in the guard catalog for the specified guard name. The same is also true when the catalog entry for access protection via guards is created without the SECOS software product.

You must have access privileges to access the guards of another user ID. The access privileges for a guard are checked only when this guard is required for access control.

Deactivating guard protection

A guard entry is only deactivated by specifying the operand GUARDS=*NONE (in the CREATE-JV and MODIFY-JV-ATTRIBUTES commands). If the job variable does not have a basic ACL entry after that, then access protection for the job variable is only controlled by the default access control, the passwords and the retention period.

2.3.3 Default protection (user defined default values)

Specifying values for the protection attributes of job variables should correspond to commonly prescribed patterns. With the “default protection” SECOS function the user can define his own default values that can then be used instead of the default system values. These default values can be specified for a certain user or globally for all pubsets. They are stored in attribute guards (guards used to specify default values for protection attributes).

Default protection values can be set via the ADD-/MODIFY-DEFAULT-PROTECTION-ATTR command for the following protection attributes of job variables:

Protection attribute	**Meaning**
ACCESS	Default access control (type of access)
USER-ACCESS	Default access control (access for external users)
BASIC-ACL	Basic access control
GUARDS	Access control via guards
READ-PASS-WORD	Read password
WRITE-PASSWORD	Write password

Table 9: Protection attributes of job variables

Each attribute guard is linked to a rule container that contains the rules that determine to which job variable names these rules apply. See also the SECOS command ADD-/MODIFY-DEFAULT- PROTECTION-RULE in the “SECOS” [10] manual. Default protection is activated by creating a rule container.

Commands and macros to use default protection for job variables

Default protection values for protection attributes are implemented with the command CREATE-JV PROTECTION=*STD (see the “Commands” manual [1]) or the macro CATJV ..., PROTECT= DEFAULT (see page 92). The *STD or DEFAULT specifications are the default settings.

If default protection is not activated or /CREATE-JV PROTECTION-ATTR=*STD or CATJV ...,PROTECT=STD is specified, then the default system values apply when a job variable is created (first protection attribute entry). You cannot use the default protection values for existing job variables. The retention period (EXPIRATION-DATE) cannot be affected by the first entry with default protection. It is implicitly set to *TODAY by default.

The default protection cannot be applied to temporary job variables.

The following are the effects of the PROTECTION-ATTR operand (CREATE-JV command) or PROTECT operand (CATJV macro) on the default values of protection attributes for job variables:

- PROTECTION-ATTR=*STD
  The default system values are used for the default system values of individual protection attributes (see table 10) if not explicitly specified otherwise.
- PROTECTION-ATTR=*BY-DEF-PROT-OR-STD
  - Default protection is active: Use the default protection values (from the attribute guard)
  - Default protection is not active: Use the default system values (see table 10)

Protection attribute	**Default system value**
ACCESS	WRITE
USER-ACCESS	OWNER-ONLY
BASIC-ACL	NONE
GUARDS	NONE
READ-PASSWORD	NONE
WRITE-PASSWORD	NONE

Table 10: Default system values for job variable protection attributes

Example

```
/add-default-protection-attr guard-name=defprot,user-access=*all-users    (1)
/show-default-protection-attr d* ────────────────────────────────────────  (2)
%---------------------------------------------------------------------------
%GUARD :20SG:$USER1.DEFPROT                           DEFAULT PROTECTION
ATTRIBUTES
%---------------------------------------------------------------------------
%                  % SCOPE: CREATE-OBJECT       % SCOPE: MODIFY-OBJECT-ATTR
%                  % -------------------------- % -------------------------
%ACCESS            % *SYSTEM-STD                % *SYSTEM-STD
%USER-ACCESS       % *ALL-USERS                 % *ALL-USERS
%BASIC-ACL         % *SYSTEM-STD                % *SYSTEM-STD
%GUARDS            % *SYSTEM-STD                % *SYSTEM-STD
%READ-PASSWORD     % *SYSTEM-STD                % *SYSTEM-STD
%WRITE-PASSWORD    % *SYSTEM-STD                % *SYSTEM-STD
%EXEC-PASSWORD     % *SYSTEM-STD                % *SYSTEM-STD
%DESTROY-BY-DELETE % *SYSTEM-STD                % *SYSTEM-STD
%SPACE-RELEASE-LOCK % *SYSTEM-STD               % *SYSTEM-STD
%EXPIRATION-DATE   % *SYSTEM-STD                % *SYSTEM-STD
%FREE-FOR-DELETION % *SYSTEM-STD                % *SYSTEM-STD
%---------------------------------------------------------------------------
%GUARDS SELECTED: 1                                          END OF DISPLAY
```

```
/add-default-protection-rule rule-cont-guard=sys.udj,prot-rule=1stcharequj,
     protect-obj=*par(name=j*,attrib-guard=defprot) ──────────────────── (3)
/show-default-protection-rule ─────────────────────────────────────────── (4)
%-----------------------------------------------------------------------------
%RULE CONTAINER :2OSG:$USER1.SYS.UDJ             USR ACTIVE  DEFAULT
PROTECTION
%-----------------------------------------------------------------------------
%1STCHAREQUJ    OBJECT     = J*
%               ATTRIBUTES = $USER1.DEFPROT
%               USER-IDS   = *ANY-USER-ID
%-----------------------------------------------------------------------------
%RULE CONTAINER SELECTED: 1                                    END OF DISPLAY

/create-jv jvdefprot ──────────────────────────────────────────────────── (5)
/show-jv-attributes nodefprotjv,inf=*all ──────────────────────────────── (6)
%0000000 :2OSG:$USER1.JV.DEFPROT
% USER-ACC   = ALL-USERS   ACCESS     = WRITE
% CRE-DATE   = 2010-08-13  EXPIR-DATE = 2010-08-13
% CRE-TIME   =   14:38:42  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/create-jv nodefprotjv ────────────────────────────────────────────────── (7)
/show-jv-attributes nodefprotjv,inf=*all ──────────────────────────────── (8)
%0000000 :2OSG:$USER1.NO.JVDEFPROT
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2010-08-13  EXPIR-DATE = 2010-08-13
% CRE-TIME   =   14:41:00  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES

/create-jv jvstd,protection-attr=*std ─────────────────────────────────── (9)
/show-jv-attributes jvstd,inf=*all ───────────────────────────────────── (10)
%0000000 :2OSG:$USER1.JV.STD-PROT
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2010-08-13  EXPIR-DATE = 2010-08-13
% CRE-TIME   =   14:42:04  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
```

(1) The attribute guard “DEFPROT” is created and the default value for the ACCESS protection attribute is also specified.

(2) The default values for the protection attributes are displayed. The values that apply when a job variable is created are displayed in the “SCOPE: CREATE OBJECT” column. USER-ACCESS “*ALL-USERS” is entered for the protection attribute as a user specific value. The default system values apply to all other protection attributes (the “SCOPE: MODIFY-OBJECT-ATTR” column does not apply to job variables, it only applies to files).

(3) The default protection rule “1STCHAREQUJ” is added to the rule container “SYS.UDJ” (J stands for job variable). You specify via `"PROTECT-OBJ=*PAR(NAME=j*,ATTRIB-GUARD=DEFPROT"` that this rule applies to all job variables whose name begins with J and that the job variables for which this rule applies are to obtain their default values from the “DEFPROT” attribute guard.

(4) The properties of the rule container and the rules it contains are displayed.

(5) The job variable “JV.DEFPROT” is created. Its name abides by the rule “1STCHAREQUJ” and therefore the default values are taken from the “DEFPROT” attribute guard.

(6) You can see that the values came from the “DEFPROT” attribute guard in the USER-ACCESS attribute: The value “ALL-USERS” is not a default system value.

(7) The job variable “NO.DEFPROT” is created. Its name does *not* abide by the rule “1STCHAREQUJ” and therefore the default system values are used as the default values.

(8) You can see that the default values are default system values in the USER-ACCESS attribute: The value “OWNER-ONLY” is a default system value.

(9) The job variable “JV.STD-PROT” is created. Its name abides by the rule “1STCHAREQUJ”, but the use of the default system values is forces by specifying "`PROTECTION-ATTR=*STD`".

(10) You can see that the default values are default system values in the USER-ACCESS attribute: The value “OWNER-ONLY” is a default system value.

Default protection and monitor job variables

The default protection also applies to monitor job variables. If a protection attribute is set by the default protection that prevents further access (e.g. using system commands), then the default protection setting must be changed by the user. For example, in the MONJV handler a new monitor job variable to be created is only set to USER-ACCESS=*ALL-USERS when default protection does not explicitly set USER-ACCESS=*OWNER-ONLY.

2.3.4 Particularities when monitoring and controlling jobs

Job monitoring

Job variables which monitor a job are protected against deletion and modification of protection attributes. This protection exists from the time of declaration as a monitoring job variable until termination of the monitoring function.
The monitoring function ends for an individual job when the job is terminated. For the follow-up job of a repeat job the monitoring function begins again with the LOGON of the follow-up job and ends when the job is terminated.
In the case of a calendar job, the monitoring function ends only when the entire calendar job has been terminated.
A privileged user (TSOS privilege) can also terminate monitoring of a job prematurely with the MODIFY-JOB-OPTIONS command or transfer it to another job variable. In this case protection ends with the loss of the monitoring function.

The protection is indicated in a special output line by means of the SHOW-JV-ATTRIBUTES command. If you enter SELECT=*BY-ATTRIBUTES(MONJV=*YES) the job variables which are protected in this way are displayed. The protection can be explicitly canceled via MONJV-PROTECTION in the MODIFY-JV-ATTRIBUTES command. if protection is removed, then the job variable monitoring the job cannot be supplied with current values. Some components (e.g. JMS) therefore set the protection at each access.

Conditional job control

Job variables which are used in conditional commands or macros during conditional job control are protected against deletion and against modification of the protection attributes.

The protection is not displayed separately in the output of the SHOW-JV-ATTRIBUTES command. However, when SELECT=*BY-ATTRIBUTES(CJC-PROTECTION=*YES) is specified, the output can be reduced so that only those job variables are displayed for which this protection exists.

The protection lasts for as long as a job variable is used in a condition (e.g. from the activation of the CJC command sequence until its termination, or until the sequence is deleted, or until the wait time of a WAIT command has expired).

2.4 Loss of access to a job variable

The following table indicates how the system responds if it is not possible to access a job variable that is in use.

S i t u a t i o n		
The job variable is in the catalog of a different processor, the connection to which has been interrupted. The catalog was entered in the MRSCAT of the home computer by means of: ADD-MASTER-CATALOG-ENTRY or via MODIFY-MASTER-CATALOG-ENTRY and the following operands:	The job variable is in a catalog which was exported by means of EXPORT-PUBSET. The following operands were specified:	

DIALOG-WAIT= value
BATCH-WAIT= value
	TERMINATE-JOBS= NO	TERMINATE-JOBS = YES
		or additionally a FORCE-PUBSET-EXPORT

R e s p o n s e		
The command (or also the macro) which wants to access the catalog waits the specified time, then execution is aborted with an error (spin-off). A wait time with the value zero results in immediate abortion. Without error handling, this results in termination of the procedure or batch job. If error handling take place, the next catalog entry once again leads to the defined wait time.	A current job or the program can terminate the access	A current job is terminated immediately
Default: value=30 seconds value=28800 seconds	The catalog is not available for jobs still being initiated (termination)	

Table 11: System response to failure to access a job variable

Further information on the use of job variables in a multiprocessor network, along with possible situations precipitating a loss of access, can be found in the manual “HIPLEX MSCF” [8].

2.5 Values of the user job variables

Any combination of characters, either in character or hexadecimal form, can be assigned by the user as the value for a job variable. This information must not exceed a length of 256 characters (bytes). It is possible to address the entire length of the job variable or, if only a certain part of the job variable is needed, to address this part by specifying subareas. A subarea of the job variable is defined by specifying the position of the first character and the length of the subarea.

In commands, the job variable values are specified in the same format as character and hexadecimal constants in Assembler language, e.g. C'ABC' or X'C1C2C3'.
Hexadecimal constants of uneven length are padded on the right with a binary zero.

Note

When character constants are entered, a distinction is made between uppercase and lowercase letters.

This should be noted particularly when using conditional expressions in job control (see page 82) and when entering parts of commands from job variables (see page 70).

2.6 Saving user job variables

The data backup functions in BS2000/OSD-BC and in the software products ARCHIVE and HSMS also support the data backup of job variables. If job variables are lost (e.g. accidental deletion or overwriting), the job variables concerned can be reconstructed using a backup copy.

Pubset backup with snapsets

A snapset is a backup copy of an SF or SM pubset based on snap units. Each snap unit of the snap set is a backup of the assigned original disk of the pubset. Snapsets are used as pubset backups against failure of files and job variables and are generated during pubset operation by system support or the HSMS administrator and later deleted. The snapsets can be used as a logical backup of all files and job variables of a pubset. The latest snapset can also be used in the same way as a physical backup for pubset restoration on a disk basis.

For detailed information on pubsets refer to the Introductory Guide to Systems Support [3].

The command LIST-JV-FROM-SNAPSET provides information about job variables saved in a snapset (in the same way as SHOW-JV-ATTRIBUTES provides information about job variables from the current file catalog). The command is described in the “Commands” manual [1].

If job variables are lost (e.g. accidental deletion or overwriting), the job variables concerned can be reconstructed on the basis of a snapset:

- The LIST-JV-FROM-SNAPSET command provides information on job variables which were backed up on a snapset (such as SHOW-JV-ATTRIBUTES, which supplies information on job variables from the current file catalog).
  At program level, a query using the LJFSNAP macro is possible.
- The RESTORE-JV-FROM-SNAPSET command restores job variables to the status of a particular snapset or to the most up-to-date status on the basis of all existing snapsets. Renaming takes place here using the NEW-JV-NAME operand. The other options for overwriting (REPLACE option) and controlling log output (OUTPUT, REPORTING operands) are available in the same way as when restoring files.
  At program level, reconstruction using the RJFSNAP macro is possible.

A detailed description of the commands and macros is provided in the “Commands” [1] and “DVS Macros” manuals [5].

The command RESTORE-JV-FROM-SNAPSET restores job variables to the state of a particular snapset or to the latest state based on all available snapsets. Renaming is via the NEW-JVNAME operand here.

Further options for overwriting (REPLACE operand) and controlling the logging output (OUTPUT and REPORTING operands) are available as for restoring files. The command is described in the “Commands” manual [1].

Data backup with ARCHIVE and HSMS

Along with the saving of files, saving of user job variables can also be performed in the automatic backup runs of a computer center by systems support or by the HSMS administrator by means of ARCHIVE or HSMS.
In addition, users can also save their own job variables by means of ARCHIVE or HSMS.

If job variables are lost (e.g. accidental deletion or overwriting), the job variables concerned can be reconstructed on the basis of a backup copy.

A detailed description of the necessary statements can be found in the manuals “ARCHIVE” [11] and “HSMS” [12].

2.7 Special job variables

Special job variables are items of system information which the user can read like job variables. Special job variables can be addressed via the dummy user ID SYSJV. No genuine user ID $SYSJV may therefore exist in the system.

There is no catalog entry for special job variables. The path name structure cannot therefore be used. A catalog ID must not be specified.
The names of special job variables are displayed in the form “$SYSJV.<jvname>”.

In the command SHOW-JV-ATTRIBUTES, the entry “JV-NAME=$SYSJV.<jvname>” can also contain wildcards. Specifying “$SYSJV.” causes the names of all special job variables available in the system to be output.
With INFORMATION=*ALL-ATTRIBUTES, the format of special job variables is desribed in an additional output line. The description appears in the currently set task language: English or German (this can be set task-specifically with the command `/MODIFY-MSG-ATTRIBUTES`; see the “Commands” manual [1]).

Special job variables can only be read. Their use is possible only in the following commands:

- MODIFY-JV in the SET-VALUE operand (value transfer)
- MODIFY-JV-CONDITIONALLY in the SET-VALUE operand (value transfer)
- SHOW-JV in the JV-CONTENTS operand (value output)
- SKIP-COMMANDS in the CONDITION operand (in conditional expressions)

Special job variables can also be used in inputing command sections from job variables (see section “Input from job variables” on page 70).

Access to a special job variable is only possible if the desired item of information also exists in the system. For example, no program name can be output if the interrogating job has not loaded a program ($SYSJV.PROGNAME). In this case the user receives message JVS04B2 indicating that the desired special job variable is “empty” (if zero characters is possible in the following output format). The following two tables list all special job variables. The first table is sorted alphabetically and the second table (starting on page 49) is sorted by category. The second table does not contain descriptions of the special job variables.

Special job variables sorted alphabetically

Name	Output format	Contents
$SYSJV.ACCOUNT	1-8 characters	Account number of the running task
$SYSJV.COUNTER	4 characters, 0001-9999	Task specific counter [1]
$SYSJV.DATE	yy-mm-ddiii	Current date (ISO format) and the day (iii) in the current year
$SYSJV.DATE4	yyyy-mm-ddiii	Current date (ISO4 format) and the day (iii) in the current year
$SYSJV.DATE-ISO4	yyyy-mm-dd	Current date (ISO4 format)
$SYSJV.DATE-TIME-LONG	yyyy-mm-dd hh:mm:ss	Current date and time (format for text output)
$SYSJV.DATE-TIME-SHORT	yyyy-mm-dd.hhmmss	Current date and time (format for file names)
$SYSJV.DATUM	dd.mm.yyyy	Current date (German format)
$SYSJV.DAY	MON, TUE, WED, THU, FRI, SAT, SUN	Current day of the week, English
$SYSJV.HOME-CATID	1-4 characters	Catalog ID of the home pubset
$SYSJV.HOST	1-8 characters	BCAM name of the local host, e.g. D046ZE08
$SYSJV.JOB-CLASS	1-8 characters	Job class of the current task
$SYSJV.JOB-ELAPSED-TIME	-dddddddddd-hh:mm:ss	Time passed since the start of the job (see $SYSJV.JOB-LOGON), where “d” is the number of days; the number of seconds is not exact, as the time of the job start is only shown to the nearest minute: e.g. with a start time of 12:31:34 the value after 20 Seconds is -0000000000-00:00:54
$SYSJV.JOB-LOGON	yyyymmdd.hhmm	Time of the start of the job, e.g. 20010112.0901
$SYSJV.JOB-LOGON-ISO	yyyy-mm-dd hh:mm	Job start time (ISO format) e.g. 2000-01-12 09:01
$SYSJV.JOB-MONJV	0-54 characters	Name of the job monitoring job variable MONJV
$SYSJV.JOBNAME	0-8 characters	Job name of the running task

Table 12: Special job variables sorted alphabetically (part 1 of 4)

Name	Output format	Contents
$SYSJV.JOB-SOURCE	1-54 characters	Path of the original source file (ENTER- or procedure file, also displayed in the field *ORIGFILE* in /SHOW-JOB-STATUS)
$SYSJV.JOB-SPOOLIN	yyyymmdd.hhmm	Time of the job spoolin e.g. 20012212.0901
$SYSJV.JOB-SPOOLIN-ISO	yyyy-mm-dd hh:mm	Time of the job spoolin (ISO format) e.g. 2000-01-12 09:01
$SYSJV.MONAT	JAN, FEB, MRZ, APR, MAI, JUN, JUL, AUG, SEP, OKT, NOV, DEZ	Current month, German
$SYSJV.MONTH	JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC	Current month, English
$SYSJV.PERS-ID	0-8 characters	Personal ID for personal logons
$SYSJV.PRINCIPAL	1-256 characters	Principal name of the Kerberos identification in the dialog logon
$SYSJV.PRIO	1-3 digits	Priority of the running task
$SYSJV.PROCESSOR	0-8 characters	Processor name of the TIAM station; e.g. D046KR11
$SYSJV.PROCESSOR-APPL	0-8 characters	Processor name of the TIAM station in an application (e.g. OMNIS); e.g. D016ZE04
$SYSJV.PROC-LEVEL	1-3 characters	Procedure level
$SYSJV.PROC-SOURCE	1-54 characters	Path name of the copy of the source file (copied ENTER or procedure file, also shown in the field *CMDFILE* of /SHOW-JOB-STATUS)
$SYSJV.PROG-MONJV	0-54 characters	Name of the program-monitoring MONJV
$SYSJV.PROGNAME	0-41 characters	Name of the loaded program if this was loaded from an LMS library or as a * file
$SYSJV.REMAINING-BS2000-RUNTIME	5 characters: UNDEF, UNLIM or hh:mm	Remaining runtime which is available to the BS2000 system to shut down before the real or virtual machine is stopped; supported on SQ servers; on other BS2000 servers the value is always UNDEF.
$SYSJV.SNO	3 characters	System session number

Table 12: Special job variables sorted alphabetically (part 2 of 4)

Name	**Output format**	**Contents**
$SYSJV.STARTUPTYPE	1 characters	Startup type, e.g. Z, F, C, W, S
$SYSJV.STATION	0-8 characters	Station name of the TIAM station; e.g. DSB11243
$SYSJV.STATION-APPL	0-8 characters	Station name of the TIAM station in an application (e.g. OMNIS); e.g. BT200175
$SYSJV.STATIONTYPE	0-8 characters	Device type of the TIAM station, e.g. 9750 DDT
$SYSJV.STD-CATID	1-4 characters	Standard user catalog identifier (default pubset)
$SYSJV.SYSCMD	1-54 characters	Name of the assigned file or logical system file in parentheses
$SYSJV.SYSDTA	1-54 characters	Name of the assigned file or logical system file in parentheses
$SYSJV.SYSID	1-3 characters	System identifier [2]
$SYSJV.SYSLST	1-54 characters	Name of the assigned file or logical system file in parentheses
$SYSJV.SYSOUT	1-54 characters	Name of the assigned file or logical system file in parentheses
$SYSJV.SYSTEM-VERSION	10 characters	System version number e.g. V17.0A00pp with pp=PVLU
$SYSJV.TAG	MO, DI, MI, DO, FR, SA, SO	Current day of the week, German
$SYSJV.TASK-CPU-USED	ssssss.ssss	Current CPU time for the task since LOGON
$SYSJV.TASK-CPU-USED-LONG	ssssssssss.ssss	Current CPU time for the task since LOGON for so-called "long-running transactions"
$SYSJV.TASK-MODE	2-6 characters: BATCH, DIALOG, SYSTEM, TP	Mode of the running task
$SYSJV.TEMP-PREFIX	1 character	Prefix for temporary job variables and files, e.g. #
$SYSJV.TIME	hh:mm:ss	Current time

Table 12: Special job variables sorted alphabetically (part 3 of 4)

Name	Output format	Contents
$SYSJV.TIME-ZONE	+hh:mm i	Time difference between the local time zone and GMT, where the indicator “i” shows the current time: – W (winter≙standard time) – S (summertime)
$SYSJV.TIMESTAMP	yyyy-mm-ddhhmmss	Time stamp with date and time (UTC) in the same format as in a MONJV
$SYSJV.TOMORROW	yyyy-mm-dd	Tomorrows date
$SYSJV.TSN	4 characters	TSN of the running Task
$SYSJV.USERID	1-8 characters	User ID of the running Task
$SYSJV.VIRTUAL-HOST	0-8 characters	BCAM name of the virtual host (a value is only returned for TIAM V13.0 and higher)
$SYSJV.YESTERDAY	yyyy-mm-dd	Yesterdays date
$SYSJV.ZEIT	hh-mm-ss	Current time

Table 12: Special job variables sorted alphabetically (part 4 of 4)

1 Initialized to zero and incremented by one prior to each read access within a task; it thus contains values which are unique only within a task, e.g. for label generation.

2 A code assigned to a home pubset.

Special job variables sorted by category

Category	Name	Output format
Date/time	$SYSJV.DATE	yy-mm-ddiii
	$SYSJV.DATE4	yyyy-mm-ddiii
	$SYSJV.DATE-ISO4	yyyy-mm-dd
	$SYSJV.DATE-TIME-LONG	yyyy-mm-dd hh:mm:ss
	$SYSJV.DATE-TIME-SHORT	yyyy-mm-dd.hhmmss
	$SYSJV.DATUM	dd.mm.yyyy
	$SYSJV.DAY	MON, TUE, WED, THU, FRI, SAT, SUN
	$SYSJV.MONAT	JAN, FEB, MRZ, APR, MAI, JUN, JUL, AUG, SEP, OKT, NOV, DEZ
	$SYSJV.MONTH	JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
	$SYSJV.TAG	MO, DI, MI, DO, FR, SA, SO
	$SYSJV.TIME	hh:mm:ss
	$SYSJV.TIME-ZONE	+hh:mm i
	$SYSJV.TIMESTAMP	yyyy-mm-ddhhmmss
	$SYSJV.TOMORROW	yyyy-mm-dd
	$SYSJV.YESTERDAY	yyyy-mm-dd
	$SYSJV.ZEIT	hh-mm-ss
Job information	$SYSJV.ACCOUNT	1-8 characters
	$SYSJV.COUNTER	4 characters, 0001-9999
	$SYSJV.JOB-CLASS	1-8 characters
	$SYSJV.JOB-ELAPSED-TIME	20 characters
	$SYSJV.JOB-LOGON	13 characters
	$SYSJV.JOB-LOGON-ISO	16 characters
	$SYSJV.JOBNAME	0-8 characters
	$SYSJV.JOB-SOURCE	1-54 characters
	$SYSJV.JOB-SPOOLIN	13 characters
	$SYSJV.JOB-SPOOLIN-ISO	16 characters
	$SYSJV.PERS-ID	0-8 characters

Table 13: Special job variables sorted by category (part 1 of 2)

Category	Name	Output format
Job information (cont.)	$SYSJV.PRINCIPAL	1-256 characters
	$SYSJV.PRIO	1-3 digits
	$SYSJV.PROC-SOURCE	1-54 characters
	$SYSJV.PROGNAME	0-41 characters
	$SYSJV.STD-CATID	1-4 characters
	$SYSJV.TASK-CPU-USED	11 characters, ssssss.ssss
	$SYSJV.TASK-CPU-USED-LONG	15 characters, ssssssssss.ssss
	$SYSJV.TASK-MODE	2-6 characters: BATCH, DIALOG, SYSTEM, TP
	$SYSJV.TSN	4 characters
	$SYSJV.USERID	1-8 characters
Information on job variables	$SYSJV.JOB-MONJV	0-54 characters
	$SYSJV.PROG-MONJV	0-54 characters
	$SYSJV.TEMP-PREFIX	1 characters
System files and procedures	$SYSJV.PROC-LEVEL	1-3 characters
	$SYSJV.SYSCMD	1-54 characters
	$SYSJV.SYSDTA	1-54 characters
	$SYSJV.SYSLST	1-54 characters
	$SYSJV.SYSOUT	1-54 characters
TIAM-Information	$SYSJV.HOST	1-8 characters
	$SYSJV.PROCESSOR	0-8 characters
	$SYSJV.PROCESSOR-APPL	0-8 characters
	$SYSJV.STATION	0-8 characters
	$SYSJV.STATION-APPL	0-8 characters
	$SYSJV.STATIONTYPE	0-8 characters
	$SYSJV.VIRTUAL-HOST	0-8 characters
System data	$SYSJV.HOME-CATID	1-4 characters
	$SYSJV.REMAINING-BS2000-RUNTIME	5 characters: UNDEF, UNLIM or hh:mm
	$SYSJV.SNO	3 characters
	$SYSJV.STARTUPTYPE	1 characters
	$SYSJV.SYSID	1-3 characters
	$SYSJV.SYSTEM-VERSION	10 characters

Table 13: Special job variables sorted by category (part 2 of 2)

Exit 033 for special job variables

This exit routine is called when a special job variable that is not supported by the system is accessed. The access can be made either via the SHOW-JV and MODIFY-JV commands or using the GETJV and SETJV macros. Values for the special job variable which are destined for the person using the command can be passed in this exit routine. This makes it possible to implement separate special job variables.
The address of parameter area EX033 is passed in register 1. The JVSXJV field of the parameter area contains the name of the special job variable requested by the user.

The following information is passed to the exit routine:

R1	=	A(EX033 parameter list)
R12	=	A(TPR Program Manager)
R13	=	A(Backup area of the calling component)
R14	=	A(Indirect return)
R15	=	A(Exit routine)

Return information

Return information for the calling system component can be supplied in the parameter list. The length of the information (1-256) can be entered in the JVSXLEN field, the contents of the special job variable in the JVSXVAL field. The maincode in the standard header must be set to X'0000'.

If no exit routine has been specified or the maincode is not equal to X'0000', the following message is displayed:

```
%  JVS0472 NAME OF SPECIAL JOB VARIABLE NOT PERMITTED. CORRECT COMMAND
```

DSECT

A DSECT for the parameter list (address in register 1) can be created using the EX033 macro.

```
EX033    EX033 MF=D
EX033    MFTST MF=D,PREFIX=J,MACID=VSX,ALIGN=F,                         C
               DMACID=VSX,SUPPORT=(D,C,M,L),DNAME=VSXPL
EX033    DSECT ,
               *,##### PREFIX=J, MACID=VSX #####
JVSXMIN  EQU   1                          MIN. LENGTH
*
JVSXMAX  EQU   256                        MAX. LENGTH
*
*   end parameterarea
```

```
JVSXHDR  FHDR  MF=(C,JVSX),EQUATES=NO                    STANDARDHEADER
JVSXHDR  DS    0A
JVSXFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
*
JVSXIFID DS    0A              0   INTERFACE IDENTIFIER
JVSXFCTU DS    AL2             0   FUNCTION UNIT NUMBER
*                                  BIT 15    HEADER FLAG BIT,
*                                  MUST BE RESET UNTIL FURTHER NOTICE
*                                  BIT 14-12 UNUSED, MUST BE RESET
*                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
JVSXFCT  DS    AL1             2   FUNCTION NUMBER
JVSXFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
*
JVSXRET  DS    0A              4   GENERAL RETURN CODE
JVSXSRET DS    0AL2            4   SUB RETURN CODE
JVSXSR2  DS    AL1             4   SUB RETURN CODE 2
JVSXSR1  DS    AL1             5   SUB RETURN CODE 1
JVSXMRET DS    0AL2            6   MAIN RETURN CODE
JVSXMR2  DS    AL1             6   MAIN RETURN CODE 2
JVSXMR1  DS    AL1             7   MAIN RETURN CODE 1
JVSXFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
*
*   main return codes
JVSXSUCC EQU   0                         NO ERROR DETECTED
JVSXREJE EQU   1                         INVALID SPECIAL JV
*
JVSXJV   DS    CL54                      NAME OF THE SPECIAL JV
JVSXLEN  DS    H                         RETURNED LENGTH OF JV-VALUE
JVSXUNU  DS    XL2                       UNUSED
JVSXVAL  DS    CL256                     RETURNED VALUE OF THE SPECIAL
*                                        JV
JVSXUNU1 DS    XL2                       UNUSED
JVSX#    EQU   *-JVSXHDR
```

2.8 Job/program monitoring using job variables

The job variables concept is designed primarily for the exchange of information between the jobs of a user. With the aid of these variables, so-called job chains or job networks can be set up. Any number of interdependencies between jobs can be mapped onto job variables so that job start, monitoring of progress (job trace), job restart etc. are practically automatic. Job variables can be created, updated and interrogated on program level via the macro interface. This information can likewise be used for job control.
An additional use of job variables within BS2000 is the monitoring of job/program execution in the system. To do this, the user must define a job variable as a monitoring JV in the MONJV operand of the following commands (for more information on job monitoring see page 64).

MONJV operand in job-monitoring commands:

– ENTER-JOB
– ENTER-PROCEDURE
– PRINT-DOCUMENT
– SET-LOGON-PARAMETERS
– TRANSFER-FILE (see “openFT” [15])

MONJV operand in program-monitoring commands:

– LOAD-EXECUTABLE-PROGRAM (resp. LOAD-PROGRAM)
– RESTART-PROGRAM
– START-EXECUTABLE-PROGRAM (resp. START-PROGRAM)

Privileged users can define a monitoring job variable in the following commands:

– CLEAR-VOLUME (see “SPACEOPT” [16])
– CREATE-VM (see “VM2000” [17])
– EXPORT-PUBSET
– EXTEND-VM-MEMORY (see “VM2000” [17])
– FORCE-PUBSET-EXPORT
– IMPORT-PUBSET
– MODIFY-JOB-OPTIONS
– REDUCE-VM-MEMORY (see “VM2000” [17])
– START-SPACEOPT-JOB (see “SPACEOPT” [16])
– START-SUBSYSTEM

The operating system then assigns fixed, predefined values to this job variable at certain times. These values can be interrogated by the user within a command sequence in order to follow and, if necessary, influence the processing of the job/program.

A monitoring job variable can be used in the following commands to identify a job:

- `CANCEL-FILE-TRANSFER` (see “openFT” [15])
- `CANCEL-JOB`
- `CANCEL-PRINT-JOB`
- `CHANGE-TASK-PRIORITY`
- `CANGE-TASK-CPU-LIMIT`
- `FORCE-JOB-CANCEL`
- `HOLD-JOB`
- `HOLD-TASK`
- `MODIFY-JOB`
- `MODIFY-JOB-OPTIONS`
- `MODIFY-PRINT-JOB-ATTRIBUTES`
- `MOVE-TASK-TO-CATEGORY`
- `RESUME-JOB`
- `RESUME-PRINT-JOB`
- `RESUME-TASK`
- `SHOW-JOB-STATUS`
- `SHOW-FILE-TRANSFER` (see “openFT” [15])
- `SHOW-PRINT-JOB-ATTRIBUTES`
- `SHOW-RESOURCE-ALLOCATION`

Up to the end of the job to be monitored, the job variable entry contains the note that the job variable has a monitoring function. The job variable entry can then only be changed if the protection is canceled in the MODIFY-JV-ATTRIBUTES command in the MONJV-PROTECTION operand. If protection is deactivated, then the job monitoring job variables cannot be supplied with current values anymore. That is why some components (e.g. JMS) activate protection each time something is accessed.

2.8.1 Values for monitoring job variables

A monitoring job variable consists of a system section (bytes 1-128) and a user section (bytes 129-256).
While a job or a program is executing, specific values are set at certain times in the system section of a monitoring job variable
The fields supplied with values (system section) by the MONJV handler and the system components are protected against every other write access from the beginning of the job until the end of the job.

2.8.1.1 Job monitoring

Job monitoring job variables should, regardless of which interface is used to operate them (e.g. JMS, SPOOL, BCAM, DSSM) have a uniform structure and uniform contents to the extent that this is possible. Job monitoring in the sense of this convention are all monitoring job variables that are not monitoring programs.
The following structure is supported for the system section:

Byte	Meaning/possible values	Entered by
1-3	Status of the job: – 1st character "$" – 2nd - 3rd characters: "A" to "Z" From JV V13.0B onwards, "$XY" for example will also be possible in the future in addition to the previous values "$R", "$T", "$A", "$S" and "$M".	System components [1]
4	Contains the value null (0).	MONJV handler
5-8	TSN of the job.	System components [1)]
9-12	Catalog ID of the home pubset.	MONJV handler
13-16	Contains spaces.	MONJV handler
17	Type of MONJV: Any character from "A" to "Z" can be chosen	System components [1)]
18-20	Current system session number	MONJV handler
21-36	Start of the job (open the MONJV) in GMT time. Format: yyyy-mm-ddhhmmss	MONJV handler
37-52	Time stamp to be able to log the time of an activity in GMT time during a job. Format: yyyy-mm-ddhhmmss	User [2]
53-60	Name of the application to be monitored.	User [2)]
61-70	Reserved.	
71-128	Application specific information. Bytes 105 through 128 may possible be used by the job management system AVAS.	User [2)]

Table 14: Structure of a job monitoring job variable

[1] System components that perform the monitoring function (e.g. JMS, SPOOL, DMS).

[2] The user can supply these elements via the command MODIFY-MONJV or the macro TIMJV

The TSN and catalog ID serve to identify the monitored job when the name of its job variable is specified.
Both values are set by the corresponding system components when the job is initiated, i.e. when either LOGON or ENTER is issued.

The values for the status of the job and catalog ID are always given left-justified and filled with spaces (X'40'), while the TSN indicator is entered right-justified with leading zeros. Within the first 128 bytes (system area), all unused bytes are filled with spaces (X'40').

In SPOOL and cancel jobs, parts of the area of the first 128 bytes are required. You will find more detailed descriptions of the structure of the job variables for spool jobs in the “SPOOL” manual [13] and for cancel jobs in the “Commands” manual [1]. If the job was terminated abnormally, the status indicator of a monitoring job variable is set to the value $A. In addition, the information about the originator in abbreviated form and the comment are transferred from the TEXT operand of the CANCEL-JOB command to the system part (the first 128 bytes) of the monitoring job variable.

If the job is started for another user ID, the job variable must also be accessible to this user ID. If it is not, no status changes can be logged once the job has been accepted. The value of the status indicator remains on “$S” and the job variable is protected (MONJV protection) even though the job may have been ended.

An active monitoring job variable cannot simultaneously be assigned as a monitoring job variable to another job or program.

If access to the monitoring job variable is lost during job processing, the system's response depends on the status of the catalog containing the JV.

For reasons of job execution security it is advisable to create the monitoring job variable on the same computer on which the job itself will be processed.

Note

Specifying a temporary job variable, a JV link name or a JV subarea as a monitoring job variable is not permissible and any attempt to do so will be rejected.

Status values for interactive jobs:

Value	**Meaning/reason for setting the value**
$R	The LOGON process was successfully executed and a job was initiated.
$T	The job was terminated normally (EXIT-JOB MODE=NORMAL or LOGOFF).
$A	The job was terminated abnormally (EXIT-JOB MODE=ABNORMAL, CANCEL-JOB or system shutdown) or job monitoring was terminated prematurely or transferred to another job variable (MODIFY-JOB-OPTIONS).

Table 15: Status values for interactive jobs

Status values for batch jobs:

Value	**Meaning/reason for setting the value**
$S	The job was entered in the job queue as a result of an ENTER-JOB command after a successfully completed “SPOOL-IN”. In the case of the follow-up job of a repeat job, this status does not occur because its monitoring begins only with the LOGON.
$R	The job was selected from the job queue and started.
$M	The job was exported from the current job pool with the MOVE-JOBS command
$T	The job was terminated normally (EXIT-JOB MODE=NORMAL or LOGOFF).
$A	The job was terminated abnormally (EXIT-JOB MODE=ABNORMAL, CANCEL-JOB or system shutdown) or job monitoring was terminated prematurely or transferred to another job variable (MODIFY-JOB-OPTIONS).

Table 16: Status values for batch jobs

Examples

The /SHOW-JV command causes the contents of the job monitoring job variable MON.JOB to be output. The letter J (byte 17) means that the job variable MON.JOB is being used to monitor a **j**ob.

```
/SHOW-JV JV=MON.JOB
```

Output:	%	$R	0	1ULW	10SB	J	068	2010-11-14161559	2010-11-14161807	Appl	Info
Starting at position		1	4	5	9	17	18	21	37	53	71

The /SHOW-JV command causes the contents of the job monitoring job variable MON.SPOOL to be output. The letter S (byte 17) means that the job variable MON.SPOOL is being used to monitor a **s**pool job. For an explanation of all further and additional field contents, refer to the “SPOOL” manual [13].

```
/SHOW-JV JV=MON.SPOOL
```

Output:	%	$S	0	1ULW	10SB	S	068	2010-11-14161559	2010-11-14161807	Appl	Info
Starting at position		1	4	5	9	17	18	21	37	53	71

2.8.1.2 IMPORT-PUBSET/EXPORT-PUBSET monitoring

The functions “importing” and “exporting” of public volumes (pubsets) and catalogs are initiated by means of the IMPORT-PUBSET and EXPORT-PUBSET commands by **privileged** users (see page 237). In this case a separate job is created for the required processing. The processing status can be monitored by means of job variables. Only the status indicator is supplied with values.
Information on the effects in an MSCF multiprocessor network can be found in the manual “HIPLEX MSCF” [8].

Status values for IMPORT-PUBSET:

Value	Meaning/reason for setting the value
$I	Import job initiated successfully.
$R	Pubset imported successfully.
$A	Import terminated abnormally.
$W	For import of a shared pubset, the processor waits for confirmation by the master processor.

Table 17: Status values for IMPORT-PUBSET

Status values for EXPORT-PUBSET: Monitoring for shared pubset operation

Value	Meaning/reason for setting the value
$E	Export job started successfully.
$T	Export of a pubset completed successfully.
$A	Export terminated abnormally or with CANCEL-PUBSET-EXPORT.

Table 18: Status values for EXPORT-PUBSET

Using HIPLEX MSCF allows concurrent shared access to a pubset across a number of computers. One of the network partners (“sharers”) is nominated as the owner processor (“master”) of this shareable pubset, and becomes responsible for the administrative tasks of the network.
On the home pubset of each sharer there is a shared pubset-specific job monitoring job variable for each shared pubset; this is created when the pubset is imported (if it does not already exist).

This monitoring job variable can take on the following status values:

Value	**Meaning/reason for setting the value**
$R	Shared pubset available. Master switchover successfully completed.
$T	Shared pubset is no longer monitored; it will be or has already been exported.
$A	Master switchover terminated abnormally, e.g. because master has crashed and nomination of a new master from the slave processors has failed.
$C	Due to master crashing or being shut down, a master switchover has been performed on the public volume.

Table 19: Status values in shared-pubset mode

2.8.1.3 Program monitoring

Program monitoring job variables are *not* protected against overwriting. However, the first 128 bytes should not be used here either for user purposes as a system entry can be overwritten or this part may be overwritten by the system when the status changes.

The operating system sets the following two values in program-monitoring job variables:

Status indicator	length: 3 characters
Return code indicator	length: 4 characters

The status indicator can have the value “$R”, “$T” or “$A”.

Value	**Meaning/reason for setting the value**
$R	The program has been started/loaded or restarted.
$T	The program has been executed successfully (macro TERM MODE=NORMAL).
$A	The program has been terminated prematurely by a program error or a predefined error exit (macro TERM MODE=ABNORMAL) or aborted by a CANCEL-PROGRAM command.

Table 20: Status values for program monitoring

The return code indicator serves to transfer a return code defined by the user on program level to the job control level. The indicator can be set in the program by means of the TERM macro. The value is stored left-justified. The default value is ␣␣␣␣ for error-free termination.

Example

The command causes the contents of the program-monitoring job variable MONITOR to be output. The letter P (byte 17) means that job variable is being used to monitor a **p**rogram.

```
/SHOW-JV JV=MONITOR
```

Output:	%	$R		P	
Starting at position		1		17	

2.8.2 Identifying jobs by means of monitoring job variables

An important function of monitoring job variables is to identify jobs in the commands for job management (see page 54 for the command list).

In these commands, jobs can be addressed either via the TSN or via a MONJV (monitoring job variable).

Some of these commands (SHOW-JOB-STATUS, CHANGE-TASK-PRIORITY, CANCEL-JOB and FORCE-JOB-CANCEL) can be issued on a multiprocessor basis, i.e. they are effective across more than one computer.
Defining a monitoring job variable in a computer network is useful because the task sequence number (TSN) of a job is normally not unique in an MSCF network. Using a monitoring job variable that is unique network-wide can help identify a job wherever it is running in the MSCF network (see also the manual “HIPLEX MSCF” [8]).

In this case there is no change in the processing characteristics when entering the monitoring job variable. In particular, users cannot reference other users' jobs, even if they can access the others' monitoring job variable.

2.8.3 Protecting monitoring job variables

Job variables which are used as monitoring job variables can also be protected against access like other job variables. In particular, the user can freely allocate passwords for extended read and write protection.
The system cannot bypass these protection mechanisms when values are to be assigned to the monitoring job variables. This means that each user is responsible for ensuring that his or her job variables are accessible. Passwords must therefore be known to the system when the first access operation is performed (e.g. ADD-PASSWORD command).
If the monitoring job variable cannot be assigned, the job or program is not started.

When monitoring a job or a program, write accesses to monitoring job variables are necessary in order to set the appropriate values. If access authorization is granted when a monitoring JV is first accessed, this access right remains assigned to the system for as long as the job variable is used for monitoring.

This inherited access authorization is restricted to password protection (READ-PASSWORD/WRITE-PASSWORD).
Changing the protection attributes (e.g. shareability or BASIC-ACL) of job monitoring job variables is only possible when MONJV protection is deactivated.

The system protects the first 128 bytes (system area) of a job-monitoring job variable against write access. Certain fields in the system section (see table 14 on page 55) can still be changed with the command MODIFY-MONJV or the macro call TIMJV.
The JV entry is also protected against modification. This protection begins with ENTER-JOB, LOGON or SET-LOGON-PARAMETERS and is canceled at the end of the monitored job. If this cancellation is not possible, the JV remains locked until the next system initialization or until explicitly released by the user:

```
/MOD-JV-ATTR JV-NAME=jvname,PROTECTION=*PAR(MONJV-PROTECTION=*NO)
```

For the duration of this retention period, the JV cannot be assigned as a monitoring JV to any other job or program.
During system initialization, access to monitoring job variables is required in the following instances:

– A monitored job is removed from the job queue. The monitoring job variable is set to $A, write protection is canceled.

– A job was started by means of RERUN-AFTER-CRASH=YES or FLUSH-AFTER-SHUTDOWN=YES and is still in the job queue. The monitoring job variable remains protected.

To ensure that the values in monitoring job variables are correct, the variables must be accessible when the system is initialized. Since more than one pubset can be operated simultaneously (for details see the manual "Introductory Guide to Systems Support" [3]), the following should be noted:

A monitoring job variable should reside on the home pubset of the computer on which the job is executed. Other pubsets cannot be accessed at system initialization time.

If a BS2000 session is terminated abnormally, monitoring job variables indicate the status of the monitored job or program at the time the last entry was made.

Notes

– Note the following when monitoring *repeat jobs*:

  The MONJV contains the TSN and the job status of the first job. The TSN and the job status of a subsequent repeat are not updated until the point in time at which the repeat job is started, i.e. at LOGON time. It is not possible to interrogate the status "$S" for repeat jobs.
  This should also be noted when using the MONJV for the identification of jobs (e.g. CANCEL-JOB). It is necessary to consider whether the job currently running or already terminated or the repeat job already in type 1 is to be referenced. This repeat job can, however, only be referenced through the MONJV if it has been started.
  In addition, the MONJV is not protected from LOGOFF to the start of the repeat job. The danger of loss of access in the event of long delays between two repeats is correspondingly high.

– By contrast, the following applies to *calendar jobs*:

  The MONJV is protected for the entire runtime, i.e. also for follow-up jobs of type 1. The TSN does not change throughout the runtime. From termination of one job to the start of the next, the MONJV contains the termination status $T or $A of the predecessor.

2.8.4 Link names of monitoring job variables

When a job variable is defined as a monitoring job variable, a JV-LINK entry is automatically created with a default link name. Further use of the default link name within the monitored job or the job calling the program overwrites the entry. Access to the monitoring job variable via the default link name is thus lost.

Job monitoring:

A job-monitoring job variable (specified in the command ENTER-JOB, ENTER-PROCEDURE, PRINT-DOCUMENT, SET-LOGON-PARAMETERS or TRANSFER-FILE) is entered in the JV-LINK table of the job to be monitored with the default link name **SMONJVJ** during LOGON processing. The monitoring job variable in commands or macros within the job can be referenced by means of this link name.
Use of a temporary job variable is not possible.

Program monitoring:

A program-monitoring job variable is entered in the JV-LINK table of the calling job when a program is started or loaded with the default link name **SMONJVP**. The monitoring job variable in commands within the job can be referenced with this link name. The program can also access the job variable via the link name. The JV-LINK entry exists from the beginning of monitoring up to the end of the program (TERM or CANCEL-PROGRAM command). Use of a temporary job variable is possible, but other jobs cannot access it.

2.9 Conditional job control

Conditional job control (CJC) uses job variables and the changes in their values for event-driven job control. The user can refer to the contents of job variables in conditional commands. If changes are made to a given job variable (an event occurs), the system notifies all the jobs in which this JV is used in conditional commands. Even replacing a value by the same value counts as a change. If the specified condition is satisfied by the occurrence of an event, the desired CJC function is executed.

A complex control structure for job chains and job networks can be implemented with the aid of the functions described below.

2.9.1 Conditions and events

The user can formulate a condition with job variables and constants in the ADD-CJC-ACTION, SKIP-COMMANDS and WAIT-EVENT commands, and in the ONEVT macro. Simple conditions can be combined to form complex conditions by means of logical operators.
The effect of the command depends on the status of the condition in each case (“satisfied” or “not satisfied”). Changes in this condition status are caused by events, i.e. changes in the values of the job variables specified there. The system informs the appropriate job of every event relevant to the condition (e.g. a MODIFY-JV command for a job variable used in the condition). The condition is evaluated and, if it is “satisfied”, the actions provided for this case are carried out. The value change can be carried out by any job which possesses the appropriate access rights for the user job variable. Special job variables which are only permissible in conditions of the SKIP-COMMANDS command can only be changed by the system (e.g. date or time).

The order in which the user job is informed of events determines the order in which the events are processed within the job (i.e. the order of condition analyses). It is not possible to inform jobs of the creation of job variables. This means that conditional commands whose job variables do not yet exist are rejected. On the other hand, existing conditional commands or macros become invalid if the catalog containing the JV is definitively exported, or if the conditions are explicitly deleted by means of the REMOVE-CJC-ACTION command or the DONEVT macro.

2.9.2 Synchronizing events

Whenever a job changes a number of job variables in succession, these changes are listed in order of occurrence and analyzed in the same order by all the jobs affected. In a multi-processor network, unsynchronized value changes to the same job variable caused by a number of different jobs are not automatically processed in the same order by all the affected jobs in all the CPUs. If required, the user himself must synchronize these value changes using the MODIFY-JV-CONDITIONALLY or WAIT-EVENT command.

2.9.3 Commands/macros for conditional job control

SHOW-CJC-STATUS

The user can use this command to obtain information on jobs with currently active applications of the conditional job control (CJC). The interrogation may be formulated for processors, catalogs or job variables.

SKIP-COMMANDS

This command causes a branch within the command sequence, depending on the analysis of a condition specified as an operand.

If a SKIP-COMMANDS occurs in a command sequence, the values of all the job variables specified in the conditional operand are read immediately and the conditional expression is analyzed. If the result of the analysis is “condition satisfied”, a branch is made to the specified point in the command sequence. It is possible to branch to points in the command sequence which come before or after the SKIP-COMMANDS command. If the analysis result is “condition not satisfied”, processing continues with the command following the SKIP-COMMANDS command.

WAIT-EVENT

The WAIT-EVENT command is used to place a job in the wait state until either a specified condition has been satisfied or a predefined time interval has elapsed. In the latter case the user can use a label to specify the point at which the job is to be resumed after the interval has elapsed. The branch destination may be located before or after the WAIT-EVENT command in the command sequence. If no branch destination is specified, a branch is made to the next SET-JOB-STEP, EXIT-JOB, LOGOFF, END-PROCEDURE or IF-BLOCK-ERROR in the command sequence.

ADD-CJC-ACTION

By means of the ADD-CJC-ACTION command a job can wait repeatedly in asynchronous operation within a given time interval for a specified condition to be satisfied, while continuing normally with the processing of the next command sequence. Different actions can be defined for the two cases, “condition satisfied” and “timeout”.
Actions that are permitted include starting jobs (ENTER-JOB), starting procedures (ENTER-PROCEDURE) and setting job variables (MODIFY-JV)

A CJC command sequence begins with the ADD-CJC-ACTION command and ends with the END-CJC-ACTION command. Between these commands, two command sequences for “condition satisfied” and “timeout” may be inserted, with any number of permitted actions (ENTER-JOB, ENTER-PROCEDURE or MODIFY-JV commands). The command sequences are stored by the system for subsequent execution. If the “condition satisfied” event occurs, a command currently being processed is still executed (WAIT-EVENT wait state is interrupted immediately) and the first CJC command sequence is processed. The job is then continued normally from the point at which the interruption occurred. Similarly, the second CJC command sequence is processed if a “timeout” occurs.
If the job is in program mode (including TU contingency), it is immediately interrupted and the specified CJC command sequence is executed. A return is then made to the point at which the program was interrupted.

Note

Whenever the “condition satisfied” event occurs, the CJC command sequence is executed. If several iterations have been specified, a test is carried out before each pass to check that the condition is still true.
The time interval within which the system is to check the condition (timeout) can also be specified in the command.

The ADD-CJC-ACTION command remains effective in the system until

- the job is terminated (EXIT-JOB or LOGOFF command) or aborted (CANCEL-JOB command),
- the maximum number of “condition satisfied” events is reached,
- the time interval expires (timeout),
- the ADD-CJC-ACTION is deleted (REMOVE-CJC-ACTION), or
- a catalog containing one of the job variables involved is exported.

Note

No checkpoint can be written (WRCPT macro) while the ADD-CJC-ACTION command is effective. A restart (RESTART-PROGRAM command) terminates all ADD-CJC-ACTION commands still active.

END-CJC-ACTION

The END-CJC-ACTION command is used in the command sequence to terminate the CJC command sequence. A CJC command sequence begins with an ADD-CJC-ACTION command and ends with an END-CJC-ACTION command. All the commands which are to be executed in the case of “condition satisfied” or after expiry of the specified time (timeout) are contained between these delimiting commands.
After END-CJC-ACTION the CJC command sequence is stored under a job-related local number. Entering this number deletes a specific CJC command sequence (REMOVE-CJC-ACTION) again.

REMOVE-CJC-ACTION

When the REMOVE-CJC-ACTION command is specified, either a specific CJC command sequence or all CJC command sequences still in effect are rendered ineffective (ALL operand).
All the information stored relating to deleted CJC command sequences is deleted.

ONEVT (macro)

Users can employ the ONEVT macro to make changes in the values of job variables on program level in order to control the execution of the program. To this end a condition may be specified as the operand. Each time the value of a job variable used in the program is changed, the system checks whether the condition has been satisfied. The situation “condition satisfied” has the characteristics of an event as defined by TU eventing by way of so-called event items. This means that a corresponding event item must be created first with the ENAEI macro and either the expected event must be requested with SOLSIG or a contingency routine must be defined for it.

DONEVT (macro)

The DONEVT macro is used to cancel the ONEVT macro. Since the TIMEOUT value in SOLSIG can also be employed to stop the system waiting for an ONEVT condition (JV event), it is advisable (and logically mandatory) to use DONEVT and DISEI to prevent ONEVT having a “delayed” effect.

Otherwise, if ONEVT and SOLSIG are repeatedly executed using the same event item, this SOLSIG can be used to request a delayed event of the old ONEVT.
Alternatively, each time the ONEVT macro is called, a different value can be assigned to the POST operand in order to differentiate events of different ONEVTs and discard old ones.

Further details on the subject of eventing can be found in the “Executive Macros” manual [4].

The table below shows the application domain for conditional commands and macros.

Command / macro	**Program level**	**Command level**			
		Batch mode		**Interactive mode**	
		Proc. mode	**Com.mode**	**Proc. mode**	**Com.mode**
SHOW-CJC-STATUS	--	x	x	x	x
SKIP-COMMMANDS	--	x	x	x	--
WAIT-EVENT	--	x	x	x	x
ADD-CJC-ACTION	--	x	x	x	x
END-CJC-ACTION	--	x	x	x	x
REMOVE-CJC-ACTION	--	x	x	x	x
ONEVT	x	--	--	--	--
DONEVT	x	--	--	--	--

Table 21: Application domain for conditional commands and macros

Note

Temporary job variables in the commands/macros for conditional job control are only possible within the job which creates them.
Access to temporary JVs of another job is not possible.

2.10 Input from job variables

A job variable can replace parts of commands or statements. SDF replaces the job variable with its value prior to execution. The resulting input is checked syntactically by SDF.

Job variable replacement is possible in unguided dialog, in procedures and in batch operation. In (temporary) guided dialog, replacement is only permissible in the NEXT line and in the input for operand values.

The job variable is specified:

– directly by its name in the form "`&(jv-name)`".

– indirectly by its job variable link name in the form "`&(*jv-link)`". Prior linkage of the link name with the job variable takes place via the command
`SET-JV-LINK LINK-NAME = jv-link, JV-NAME = jv-name`.

Note

Before job variable replacement, ACS, if required, replaces the alias of a JV by its real path name (in accordance with the alias catalog entry).

The following restrictions apply to job variable replacement:

– An expression can be replaced only by a job variable in its full length.

– The job variable to be used must have read access, otherwise the input is rejected as a syntax error.

– Replacement is not possible within CJC command sequences.

– Job variables cannot be substituted for input data. SDF treats statements intended for programs with SDF interface like commands and not like input data.

– In procedures or ENTER files, job variables cannot replace the leading slash that precedes commands or the two leading slashes that precede statements, the leading period of non-S-marks, the semi-colon as command separator and continuation mark.

– Job variables cannot be used as procedure parameters.
This restriction may be circumvented, e.g. by using a link name (see Example 2 below).

– Expressions cannot be nested.

– In interactive mode, JV-REPLACEMENT=*AFTER-BUILTIN-FUNCTION is set; i.e. job variable replacement in the form shown above is performed only if no identically named S variable or built-in function is known. The replacement is performed in the order: S variable, built-in function, and, finally, job variable.
When a user or catalog ID is entered in the job variable name, a job variable is uniquely referenced.

Job variable replacement in S procedures is affected by the JV-REPLACEMENT setting in the SET-PROCEDURE-OPTIONS command in the following manner:

– In S procedures, JV-REPLACEMENT=*NONE is preset; i.e. only S variables or built-in functions are replaced in the specified order.

– The JV-REPLACEMENT setting can be changed by means of the operand of the same name in the SET-PROCEDURE-OPTIONS command in S procedures. When JV-REPLACEMENT=*AFTER-BUILTIN-FUNCTION is set, incompatibilities can be avoided if the JV name is specified with the user ID and/or catalog ID in the previous form of job variable replacement.

Job variable replacement is also possible using the built-in function JV(). In this case, it is specified in the form "`&(JV(JV-NAME=string_expression,...))`". Specification of a subarea is permissible and optional. See also built-in function JV() in the manual “Commands” [1].

Example 1: Replacement in dialog

```
/cre-jv jv=cmd          ---------------------------------------------------- (1)

/mod-jv jv=cmd,set-value='SHOW-FILE-ATTR'  ---------------------------- (2)

/&(cmd)                 ---------------------------------------------------- (3)
%         3 :2OSG:$USER1.ALT.SYS.LOGON.USERPROC.X1
%        51 :2OSG:$USER1.ALT.SYSSDF.USER.EXAMPLE.1
%        21 :2OSG:$USER1.DATEI.1
%        48 :2OSG:$USER1.DATEI.2
%        84 :2OSG:$USER1.DATEI.3
%        66 :2OSG:$USER1.OUT.SORT1-2
%         3 :2OSG:$USER1.PROC.JV
%:2OSG: PUBLIC:     7 FILES RES=      276 FRE=       39 REL=      21 PAGES

/mod-jv jv-=cmd,set-value='-FILE-ATTR FILE-NAME=PROC.'  --------------- (4)

/set-jv-link link-name=cmdlink,jv-name=cmd  --------------------------- (5)

/sh&(*cmdlink)
%         3 :2OSG:$USER1.PROC.JV    ---------------------------------- (6)
%:2OSG: PUBLIC:     1 FILE  RES=        3 FRE=        2 REL=       0 PAGES
```

(1) The job variable name 'CMD' is created.

(2) The job variable CMD contains the value “SHOW-FILE-ATTR”.

(3) After the command has been sent, the variable string is replaced by the command defined in the job variable and the command is executed.

Note

The job variables are replaced because no S variables or built-in function with the name “CMD” exists. If a user ID or catalog ID is used in the job variable name (e.g. “`&($user1.cmd)`” instead of “`&(cmd)`”), the job variable “CMD” is uniquely referenced.

(4) The value of the job variable CMD is changed. It now only contains a part of the command name “-FILE-ATTR” and the partially qualified file name “PROC.”.

(5) The link name “CMDLINK” is assigned to the job variable CMD.

(6) After the command has been sent, the variable string is replaced by the command part assigned to the job variable and the command is executed. The reference to the job variable is formed by the link name.

Example 2: Replacement in a non-S procedure

```
/BEGIN-PROC PAR=YES(PROC-PAR=(&PARAM1))      ———————————————————————————— (1)
    .
    .
/SET-JV-LINK LINK-NAME=PARAM1,JV-NAME=&PARAM1   ————————————————————————— (2)
/&(*PARAM1) FILE-NAME=LST.JOB           ————————————————————————————————— (3)
    .
    .
/END-PROC
```

(1) The job variable specified by the procedure parameter PARAM1 is to contain the command to be executed in each case.
Since the specification “&(&PARAM1)” is not permissible, the alternate route via a link name is taken.

(2) The current job variable name is used for the procedure parameter PARAM1 and is linked with the link name PARAM1.

(3) The contents of the specified job variable are used for the link name PARAM1. If, for example, the job variable contains the value PRINT-DOCUMENT the file LST.JOB is printed out.

Example 3: Replacement in an S procedure

S procedure *sproc*:

```
/        SET-PROC-OPT  JV-REPLACE=*AFTER-BUILTIN ──────────────────────  (1)
/        DECL-PAR      JV-1(INIT=*PROMPT)──────────────────────────────  (2)
/        &(JV(JV-NAME=JV-1))   FILE-NAME=LST.JOB ──────────────────────  (3)
/FEHL:   IF-BLOCK-ERROR
/            WRITE-TEXT  C'** Error &MC **'
/        ELSE
/            WRITE-TEXT  C'** Command &(&(JV-1)) executed **'
/        END-IF                                                 ─────  (4)

/ENDE:   EXIT-PROC
```

Calling the *sproc* procedure:

```
/create-jv jv=jv.command

/modify-jv jv=jv.command,set-val='SHOW-FILE-ATTRIBUTES'

/call-proc sproc

%JV-1: jv.command
%  DMS0533 REQUESTED FILE NOT CATALOGED IN PUBSET '1OSN'. COMMAND TERMINATED
%  SDP0004 ERROR DETECTED AT COMMAND LINE:           3 IN
PROCEDURE':1OSN:$USER1.SPROC'
** ERROR DMS0533 **────────────────────────────────────────────────────  (5)
```

(1) The JV-REPLACEMENT operand of the SET-PROCEDURE-OPTIONS command specifies that job variable replacement is allowed.

(2) Defines the procedure parameter JV-1. The value for this parameter is to be requested in interactive mode.

(3) Job variable replacement is implemented using the built-in function JV() (irrespective of the JV-REPLACEMENT setting). When the procedure is called, the expression &(JV(JV-NAME=JV-1)) is replaced by the value of the job variable whose name is transferred in the procedure parameter JV-1.

(4) If an error occurs during job variable replacement, the text “** Error” is output with the corresponding message code (&MC). If the procedure executes without error, the text of the ELSE loop is displayed. Here too, the expression &(&(JV-1)) is replaced by the value of the defined job variable. During the first step of the replacement of &(JV-1), irrespective of the JV-REPLACEMENT setting (JV-1 is uniquely an S variable), the second step of the replacement is only carried out with the setting JV-REPLACEMENT=*AFTER-BUILTIN.

(5) The requested file cannot be found in the desired pubset.

3 Commands

This chapter lists the BS2000 commands needed in connection with job variables. They are divided into the following four groups:

- job variables management
- job monitoring
- program monitoring
- conditional job control

The commands for job and program monitoring are summarized together with the relevant operands in a series of tables.

i For a complete description of the command syntax and descriptions of all the operands, refer to the “Commands” manual [1].

The functionally equivalent macros for the management of job variables are described in chapter “Macros” on page 87.

3.1 Commands for job variables management

Table of commands

Command	Macro	Function
COPY-JV	COPJV	Copy job variable
CREATE-JV	CATJV	Create new job variable
DELETE-JV	ERAJV	Delete job variable
MODIFY-JV	SETJV	Modify job variable contents
MODIFY-JV-ATTRIBUTES	CATJV	Change catalog entry for job variable
MODIFY-JV-CONDITIONALLY	CSWJV	Modify job variable contents conditionally
MODIFY-MONJV	TIMJV	Set elements in the system section of a job monitoring JV
REMOVE-JV-LINK	RELJV	Delete JV-LINK entry
SET-JV-LINK	DCLJV	Define job variable link name
SHOW-JV	GETJV	Output job variable contents
SHOW-JV-ATTRIBUTES	STAJV	Output job variable attributes
SHOW-JV-LINK	LNKJV	Output JV-LINK entry

Table 22: Commands for the management of job variables

The commands for managing job variables can be used in interactive mode or batch mode.

A complete description of the commands can be found in the “Commands” manual [1].

3.2 Commands for job monitoring

This section presents an overview of the commands.

– These are initially commands for which job monitoring by means of job variables is possible (see table 23)
– There are also commands which perform job management or provide information about a job. When a job is monitored, it can be named using the name of its monitoring job variable (MONJV) (see table 24).

A complete description of the commands can be found in the “Commands” manual [1].

3.2.1 Commands for defining job monitoring

The commands below initiate a job (only MODIFY-JOB-OPTIONS refers to an existing job) and offer the option of defining a job variable for monitoring this job in the MONJV operand. In addition, a password which is required for job variable access can be specified in the JV-PASSWORD operand.

Command	**Function**
ENTER-JOB	directs the system to execute a command sequence as an independent (batch) job (see also Note on ENTER-JOB)
ENTER-PROCEDURE	starts a command sequence stored in a procedure file, as a batch job
MODIFY-JOB-OPTIONS	modifies the parameters (logging and job monitoring) of an existing job (see also Note on MODIFY-JOB-OPTIONS)
PRINT-DOCUMENT	initiates a print job
SET-LOGON-PARAMETERS	initiates an interactive or batch job (see also Note on ENTER-JOB)
TRANSFER-FILE	transfers a file from the local to the remote system, or vice versa (see “openFT” [15])

Tabelle 23: Commands for defining job monitoring

Notes on the use of a MONJV

The job variable must not be protected against overwriting.

If the job variable cannot be accessed at the time of command processing, an error message is output to SYSOUT and the command is rejected. If the specified job variable does not yet exist under the current user ID, it is created by the system (only if the job is accepted by the system) and is made shareable.

If the job variable is protected by a password, it can be declared as a monitoring job variable only if the job submitter has already entered the password in the password table (by means of ADD-PASSWORD) or specifies it in the JV-PASSWORD operand of one of the commands listed in table 24 on page 79.
The JV-PASSWORD operand is ignored if no MONJV is declared.

Note on ENTER-JOB

When monitoring a repeat job, the MONJV can only contain one TSN and the associated job status at a time. Repetition of the repeat job (next job) can only be taken from the start time of the MONJV. This is especially important when the MONJV is to be used to identify a job in commands (e.g. CANCEL-JOB).

The status “$S” cannot be queried for job repetitions, since logging begins only at the start time (with “$R”). It should also be remembered that from the termination of the current job to the start of the next job, the MONJV is *not* protected (risk of loss of access).

Operand entries in the SET-LOGON-PARAMETERS or LOGON command at the start of the cataloged command sequence are evaluated only if the operand concerned is not specified in the ENTER-JOB operand and the job is started either at the console or by specifying DEFAULT-FROM-FILE=*YES. In all other cases a MONJV can consequently only be defined in the ENTER-JOB command.

A batch job that is to run on a remote processor can only be accessed by a MONJV if each catalog ID of the home pubset of the partner processor is entered in the MRSCAT of the processors involved.

Note on SET-LOGON-PARAMETERS

In a batch job, this is the first command in the command sequence to be processed. The batch job is submitted to the system by means of the ENTER-JOB command.

When a batch job is executed that was not started on the console or by specifying DEFAULT-FROM-FILE=*YES, the operand entries in the SET-LOGON-PARAMETERS or LOGON command are not evaluated. In this case a monitoring JV can only be declared by means of the ENTER-JOB command.

Note on MODIFY-JOB-OPTIONS

Only a privileged user (TSOS privilege) can modify the monitoring of an existing job using the MODIFY-JOB-OPTIONS command. The privileged user can terminate existing monitoring prematurely in the MONJV operand, start monitoring with the specified job variable or, in the case of existing monitoring, transfer it to the specified job variable.

3.2.2 Commands for job monitoring

The commands below manage a job or provide information. If the job is monitored by a job variable, this job can optionally be identified via its monitoring job variable (name or link name) instead of the TSN.
The specification is made in the MONJV operand (with a few exceptions, in the structure JOB-IDENTIFICATION=*MONJV(...)).

Command	Function
CANCEL-FILE-TRANSFER	cancels a file transfer job running under the current ID or accessing the current ID (see “openFT” [15])
CANCEL-JOB	cancels an interactive, batch or print job running under the current user ID (see also Note on CANCEL-JOB)
CANCEL-PRINT-JOB	cancels a print job running under the current user ID
CHANGE-TASK-PRIORITY	changes the execution priority of a job that has been started
FORCE-JOB-CANCEL	cancels the specified job without delay
HOLD-JOB	places a job that has not yet been started in the wait state
MODIFY-JOB	modifies the attributes of the job (type 1 only)
MODIFY-JOB-OPTIONS	modifies the parameters (logging and job monitoring) of an existing job
RESUME-JOB	cancels the wait state of the specified job
SET-SYSLST-READ-MARK	sets a read mark in the job’s SYSLST file
SET-SYSOUTREAD-MARK	sets a read mark in the job’s SYSLOUT file
SHOW-FILE-TRANSFER	provides information about a (see “openFT” [15])
SHOW-JOB-STATUS	provides information about a file transfer job
SHOW-RESOURCE-ALLOCATION	provides information about resources allocated to the job

Tabelle 24: Commands for defining job monitoring

The commands below are rejected if the job variable specified in the JOB-IDENTIFICATION or SELECT operand does not identify a monitoring job variable. The commands are also rejected if the monitoring job variable cannot be accessed (protection by password).

The monitoring job variable is automatically entered under the link name **SMONJVJ** in the JV-LINK table of the job being monitored. This default link name enables the job to access its monitoring job variable and the information that it contains, without needing to know the name beforehand. This is particularly significant for cataloged command sequences when multiple batch jobs are to run concurrently. It must be noted here that the standard link name is not assigned explicitly in the job’s command sequence and thus overwrites the assignment.

Note on CANCEL-JOB

If a job has been terminated abnormally, i.e. the CANCEL-JOB command has been executed with STEPS=*ALL (default setting), the status indicator of a monitoring job variable is set to the value $A. In addition, the information about the originator is transferred in shortened form, together with the comment from the TEXT operand of the command, to the system part (the first 128 bytes) of the monitoring job variable:

– the information about the originator begins at byte 37 and consists of the character string *CAN:info*, where *info* contains the first 27 bytes of the originator information supplied to SYSOUT;

– the comment begins at byte 70 and consists of the character string *TEXT:text*, where *text* contains the first 51 bytes of the comment specified in the operand.

3.3 Commands for program monitoring

This section provides an overview of commands in which job variables can be defined to perform program monitoring (see table 25 on page 81).

Command overview

Command	Function
LOAD-EXECUTABLE-PROGRAM (or LOAD-PROGRAM)	loads a program (LLM, load or object module) into memory
RESTART-PROGRAM	starts a program at its checkpoint (restart)
START-EXECUTABLE-PROGRAM (or START-PROGRAM)	loads and starts a program (LLM, load or object module)

Table 25: Commands for program monitoring

A complete description of the commands can be found in the “Commands” manual [1]. Additional information on loading and starting programs is contained in the manual “BLSSERV” [14].

Notes on the use of a MONJV

- The job variable must not be protected against overwriting.
- If a temporary job variable is used, it should be noted that only the calling job has access to the monitoring job variable.
- If the job variable does not exist, it is created.
- If the job variable exists at the time the command is entered, but cannot be accessed because of a serious error (e.g. catalog destroyed), a message is output to SYSOUT and the command is rejected.
- The default link name **SMONJVP**enables the program to access its monitoring job variable without the name needing to be known when the program is written. It should be noted that assigning the default link name a second time in the job results in this assignment being overwritten.

3.4 Commands for conditional job control

This section provides a detailed description of the rules for formulating conditional expressions and an alphabetically ordered list of the commands for conditional job control:

Conditional expressions

A conditional expression is used to make processing of the ADD-CJC-ACTION, SKIP-COMMANDS or WAIT-EVENT command dependent on job variable values. A conditional expression may be either “true” or “false”. It is composed of one or more relational expressions <relation.exp> which can be constructed as follows. Alternatives are separated by “/”; optional entries are enclosed in angle brackets.

```
<relation.exp>      ::= (<term> <comparison-op> <term>)

<term>              ::= <jvid> / <jv-subarea-def> / <const>

<relational-op>     ::=  < /  > / =  / <= / >= / <>
                        LT / GT / EQ / LE / GE / NE

<jvid>              ::= <jvname> / <*jvlink> / <#jvname> / <special-jvname>

<jvname>            ::= Name of a permanent user job variable

<*jvlink>           ::= Link name of a user job variable with
                        leading asterisk (*)

<#jvname>           ::= Name of a temporary user job variable

<special-jvname>    ::= Name of a special job variable without catalog ID
                        but with specification of user ID SYSJV (only
                        permitted in the SKIP-COMMANDS command).

<jv-subarea-def>    ::= (<jvid> [,[<start>][,<length>]])

<start>             ::= <integer 1..256>  default=1

<length>            ::= <integer 1..64>  default=64
                        (with <start>+<length> <= 256)

<const>             ::= <c-string 1..64> / <x-string 1..128>
                        e.g.:
                        C'HALLO' / 'HALLO' or
                        X'00FF'  / X'0FF'
```

The operators LT, GT, EQ, LE, GE, NE must be separated from other letters by a blank (in constants, JV names or link names).
Particular attention should also be paid to differentiating between uppercase and lowercase notation in conditional expressions.

Compound conditional expressions are produced by combining relational expressions with the aid of logical operators:

```
<cond.exp>          ::= (<relation.exp> <logical.op> <relation.exp>) /
                        (<cond.exp> <logical.op> <cond.exp>) /
                        NOT <cond.exp>)
<logical.op>        ::=   AND     /     OR    /    XOR
                         logical    logical      exclusive
                          AND          OR           OR
```

Example

```
(JV1=C'START')

((JV2=C'O.K.') AND (*LINK<=C'12'))

(((JV4,10,3)=C'NEU') OR (#TEMP.JV=C'Y') AND (*LINK2=X'00'))
```

Note

A conditional expression is evaluated “from inside to outside”, in accordance with the parentheses. On the same level of parenthesis, the logical operations are performed in the following order:

```
1. NOT
2. AND
3. OR
4. XOR
```

For example, the expression

```
(NOT (JV1=C'ABC') OR (JV2=C'Z') AND (JV3<>JV4))
```

is evaluated as

```
((NOT (JV1=C'ABC')) OR ((JV2=C'Z') AND (JV3<>JV4)))
```

When comparing job variable values, certain points must be kept in mind:

– Relational expressions are evaluated one byte at a time from left to right. The appropriate bit pattern in EBCDIC code is decisive in determining the result of the comparison, e.g. the printable digits 0 through 9 (X'F0' through X'F9') are “greater” than the letters A through Z (X'C1' through X'E9'), and uppercase letters are greater than lowercase letters.

– If the comparison value lies outside the definition range of a job variable or the job variable is empty (e.g. after a declaration), the result of the comparison is always “false”, even if the test is performed using “<>” (not equal).

– If the contents of a job variable consist solely of binary zeros, it is not empty. Furthermore, if there are two otherwise identical job variable values, the one which is one binary zero longer is considered the greater of the two.

Example

The following expressions are “true”:

```
          C' ' < C'A'
          C'a' < C'A'                    X'81' < X'C1'
          C'A' < C'B'                  X'0123' < X'0124'
          C'B' < C'BB'                   X'C1' < X'C100'
C'GUTEN ABEND' < C'GUTEN MORGEN'
  C'ZZZZZZZZZ' < C'0'      (!)     X'F0F0F0F0' < X'F1'
          C'8' < C'9'
  C'899999999' < C'9'      (!)         X'3FFF' < C' '
```

The following queries are answered with “false”.

```
(IN)     CALL-PROCEDURE NAME=PROC.JV
(IN)     /.ANF BEGIN-PROCEDURE LOGGING=*ALL
(IN)     /CREATE-JV JV=JV1
(IN)     /SKIP-COMMANDS TO-LABEL=END,IF=*JV(CONDITION=((JV1,11,1) NE '2'))
(OUT)    %  CJC0011 SKIP COMMAND: CONDITION = FALSE
(IN)     /MODIFY-JV JV-CONTENTS=*SUBSTRING
(  )               (JV-NAME=JV1,POSITION=1,LENGTH=5),SET-VALUE=C'12345'
(IN)     /SKIP-COMMANDS TO-LABEL=END,IF=*JV(CONDITION=((JV1,6,2) EQ 'A1'))
(OUT)    %  CJC0011 SKIP COMMAND: CONDITION = FALSE
(IN)     /SKIP-COMMANDS TO-LABEL=END,IF=*JV(CONDITION=((JV1,6,2) NE 'A1'))
(OUT)    %  CJC0011 SKIP COMMAND: CONDITION = FALSE
(IN)     /.END END-PROCEDURE
```

A conditional expression is rejected in the following cases:

1. A referenced JV is not accessible (catalog destroyed).
2. A referenced JV does not exist in the specified catalog.
3. The user is not authorized to access a JV which is either protected by a read password or not shareable.

Overview of commands

Command	**Function**
ADD-CJC-ACTION	Initiates a CJC command sequence
END-CJC-ACTION	Identifies the end of a CJC command sequence
REMOVE-CJC-ACTION	Ends the effectiveness of a CJC command sequence
SHOW-CJC-STATUS	Displays information about jobs currently working with conditional job control
SKIP-COMMANDS	Branches conditionally within a command sequence
WAIT-EVENT	Initiates a conditional wait state

Table 26: Commands for conditional job control

In the CONDITION operand of the ADD-CJC-ACTION, SKIP-COMMANDS and WAIT-EVENT commands, a compound condition can be specified with the aid of job variable names and values. The section “Conditional expressions” on page 82f is devoted to a detailed description of this central metaconcept “conditional expression”.

(The abbreviation CJC stands for Conditional Job Control.)

A complete description of the commands can be found in the “Commands” manual [1].

4 Macros

This chapter describes the macro interface of the job variable functions for Assembler programmers. The macros have basically the same functions as the equivalent commands. Specifically, the following macros are described here:

4.1 Overview of macros

Macro	**Function**
CATJV	Catalog job variable
COPJV	Copy job variable
CSWJV	Check and set job variable
DCLJV	Define job variable link name
DONEVT	Delete condition for job variable event
ERAJV	Erase job variable
GETJV	Get job variable value
JVSEL	Limit job variable selection of the STAJV macro to specific attributes
LNKJV	Link job variables to JV-LINK entries
ONEVT	Set condition for job variable event
RELJV	Remove JV-LINK entry
SETJV	Set job variable
STAJV	Output job variable attributes
TERM	Terminate program and procedure step
TIMJV	Supply elements in the system section of a job monitoring JV

Table 27: Overview of macros

S-type macros can use the operands MF, PREFIX, PARAM and MACID. The possible values and their meanings are described below. The individual macro descriptions specify which values are permissible or act as the default for the respective macro.

4.2 General macro operands

4.2.1 The MF operand

The MF operand determines the type of macro generation.

MF value	Meaning/effect
MF=S	Default value; first generates the instruction part, then the data area. The data area does not contain any field names. The standard header is initialized.
MF=D	Generates a data area with field names and explanatory equates. The formation of field names can be influenced by the PREFIX or MACID operands in the call. The data area begins with the DSECT statement. This type of macro expansion is referred to below as generation of a DSECT. A DSECT describes the structure of a storage area, without itself occupying storage space. The location counter is reset to zero.
MF=C	Generates a data area with field names and explanatory equates. The formation of field names can be influenced by the PREFIX or MACID operand in the call.
MF=L	Generates only the data area. The data area does not contain any field names. The standard header is initialized. The generated data area can be referenced in a call with MF=E.
MF=E	Only the instructions necessary for calling the function module are generated. The instruction part usually ends with an SVC. The address of the data area with the parameter values must be specified in the macro call. MF=(E,addr) MF=(E,(r)) Only the processing code (SVC) is generated. addr or the register denoted by r contains the address of the data area (parameter list) for the MF=L expansion. MF=E[,PARAM=addr] MF=E[,PARAM=(r)] The PARAM operand specifies the address of the operand list. r = register containing the address of the data area. The register must be loaded with this address value before the macro call is made.
MF=M	The call generates instructions which supply all explicitly specified function operands in the existing operand list with the new values.

Table 28: MF operand in macros

When the instruction part and operand list are generated separately, the values specified in the assembly parameters (VERSION, PARMOD) must be the same.

The use of a standard header is dependent on the assembly parameter VERSION. Without standard header, the SVC 133 is generated in the instruction part; with standard header, the SVC 190.

Parameter lists must be aligned on a word boundary.

The parameter list should be supplied with values only through the corresponding macro with the explicitly specified operands. Complex interdependencies often exist between the individual bits, especially in the case of CATJV. Direct amendment of the parameter list is therefore inadvisable.

During macro processing, the address of the parameter list is loaded into register R1 (with MF=E and MF=S).

4.2.2 The PREFIX operand

In the macros it is possible to specify the PREFIX operand in various forms of the expansion. This controls the generation of the symbolic names for the parameter list. The prefix consists of up to three letters.
Those macro expansions in which a prefix can be specified are indicated in the “General” section of each macro description. The option of entering an arbitrary three-letter prefix is indicated by PREFIX=pre in the macro call format.

4.2.3 The PARAM operand

The PARAM operand specifies the address of the operand list.

PARAM	is used to address the operand list.
=addr	symbolic address of the operand list.
=(r)	register containing the symbolic address of the operand list.

4.2.4 The MACID operand

The MACID operand enables the user to define the second through fourth characters of the names in a parameter list. Those macro expansions in which a MACID can be specified are indicated in the “General” section of each macro description. The option of entering an arbitrary three-letter string is indicated by MACID=macid in the macro call format.

For further information on the MF, PREFIX, PARAM and MACID operands, see the manual “Executive Macros” [4].

4.3 Notational conventions

Notation	**Meaning**	**Example**
UPPERCASE LETTERS	Uppercase letters denote constants and must be entered by the user exactly as shown.	`ACCESS=READ` The user must enter `ACCESS=READ`
lowercase letters	Lowercase letters denote variables which, on entry, must be replaced by current values, i.e. their contents may differ from case to case.	`RDPASS=password` The user must enter, for instance, `RDPASS=C' OTTO'` or RDPASS=54321 etc.
{ }	Braces enclose alternatives, i.e. one entry must be selected from the specifications enclosed. Exception: default values.	$\mathtt{ACCESS=}\left\{\begin{array}{l}\mathtt{READ}\\ \mathtt{WRITE}\end{array}\right\}$ The user must enter `ACCESS=READ` or `ACCESS=WRITE`
/	The slash denotes a choice between alternatives; it has the same function as braces.	`WRITE=YES/NO` The user must enter `WRITE=YES` or `WRITE=NO`
[ ]	Square brackets enclose options, i.e. the entries may be omitted. When a comma is enclosed between square brackets in optional entries, it need only be written if the option is used. When it is outside the brackets it must be specified even if the option is not used . (Parentheses must be entered.)	`area[,length]` The user must enter, for instance, `BADR,60` or `BADR`
<u>ABC..</u>	Underscoring denotes the default value, which is the value the system assumes if the user makes no entry (= system preset). If an operand has no default value, specification of an operand is mandatory.	$\mathtt{STATE=}\left\{\begin{array}{l}\underline{\mathtt{NEW}}\\ \mathtt{UPDATE}\end{array}\right\}$ The user must enter `STATE=NEW` or `STATE=UPDATE` (no entry implies `STATE=NEW`)

Table 29: Notational conventions for macro descriptions (part 1 of 2)

Notation	Meaning	Example
.....	Ellipses denote repetition, i.e. the preceding syntactical unit may be specified one or more times in succession.	`(protect1,..4)` The user must enter, for instance, `(ACCESS)` or `(ACCESS,EXDATE)` etc.
␣	This character denotes a blank (X' 40').	`STD␣` The user must enter '`STD` '

Table 29: Notational conventions for macro descriptions (part 2 of 2)

4.4 Description of the macros

CATJV
Catalog job variable

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **CATJV** macro creates or updates the catalog entry of a job variable.

Explicitly deactivating the default protection with CATJV ..., PROTECT=STD is only supported when the operand VERSION=4 is also specified at the same time.

Using macro versions < 4 can lead to problems in conjunction with default protection: if a JV receives the protection attribute SHARE=YES or ACCESS=READ or passwords via default protection, then the non-privileged user can set the protection attribute to SHARE=NO or ACCESS=WRITE or RDPASS=NONE or WRPASS=NONE. These specifications are not evaluated, however, as the default protection settings have a higher priority.

Macro call format and operand description

Operation	Operands
CATJV	jvname1
	[,jvname2]
	[,STATE= $\left\{\begin{array}{l}\underline{\text{NEW}}\\ \text{UPDATE}\end{array}\right\}$]
	[,PROTECT= $\left\{\begin{array}{l}\underline{\text{DEFAULT}}\\ \text{STD}\end{array}\right\}$]

Operation	Operands
CATJV (cont.)	$\text{[,ACCESS=}\left\{\begin{array}{l}\text{WRITE}\\\text{READ}\end{array}\right\}\text{]}$ $\text{[,SHARE=}\left\{\begin{array}{l}\text{NO}\\\text{YES}\end{array}\right\}\text{]}$ $\text{[,OWNERAR=}\left\{\begin{array}{l}\text{NO-ACCESS}\\\text{([}\left\{\begin{array}{l}\text{READ=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{R=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{][,}\left\{\begin{array}{l}\text{WRITE=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{W=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{])}\end{array}\right\}\text{]}$ $\text{[,GROUPAR=}\left\{\begin{array}{l}\text{NO-ACCESS}\\\text{([}\left\{\begin{array}{l}\text{READ=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{R=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{][,}\left\{\begin{array}{l}\text{WRITE=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{W=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{])}\end{array}\right\}\text{]}$ $\text{[,OTHERAR=}\left\{\begin{array}{l}\text{NO-ACCESS}\\\text{([}\left\{\begin{array}{l}\text{READ=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{R=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{][,}\left\{\begin{array}{l}\text{WRITE=}\left\{\begin{array}{l}\text{YES}\\\text{NO}\end{array}\right\}\\\text{W=}\left\{\begin{array}{l}\text{Y}\\\text{N}\end{array}\right\}\end{array}\right\}\text{])}\end{array}\right\}\text{]}$ $\text{[,BASACL=}\left\{\begin{array}{l}\text{NONE}\\\text{STD}\end{array}\right\}\text{]}$

Operation	Operands
CATJV (cont.)	[,GUARDS={ NONE | ([READ={ *NONE | readguard }] [,WRITE={ *NONE | writeguard }]) }] [,MANCLAS={ *NONE | manclas }] [,RDPASS={ NONE | password }] [,WRPASS={ NONE | password }] [,RETPD={ 0 | days }] [,MONJV={ <u>UNCHANGED</u> | NO }] [,VERSION={ <u>0</u> | 1 | 2 | 3 | 4 }] [,MF={ <u>S</u> | C | (E,..) | D | L }] [,PREFIX={ <u>IDJ</u> | pre }]

jvname1
: Fully qualified path name under which the permanent or temporary job variable is cataloged. Only systems support may specify a user ID other than their own.

jvname2
: Specifies a new name for job variable jvname1. Up to versions < 2, user ID and catalog ID must not be specified as this could indicate a change of owner or catalog. From version = 2 on, a fully qualified path name can be specified, but user ID and catalog ID must match the IDs specified in jvname1.
This operand is effective only in combination with STATE=UPDATE.

STATE	Specifies that a catalog entry is to be created for a job variable which does not yet exist, or that an existing catalog entry is to be updated.
=NEW	Is the default: a new catalog entry is to be created.
=UPDATE	Specifies that an existing catalog entry is to be updated.
	Note
	In UPDATE mode, operands not specified are not set to their default value. In UPDATE mode, the corresponding value in the JV entry can only be modified by means of the ACCESS, SHARE, RDPASS, WRPASS and RETPD operands if it has been specified explicitly.
PROTECT	Specifies where the protection attributes of the job variable to be cataloged and whose value is not specified explicitly are to be taken from. The specification of this operand is only permitted together with VERSION=4.
	See the table “Default system values for job variable protection attributes” on page 37 for the value assignments.
=DEFAULT	The default setting: The protection attributes of the new job variable are cataloged with the values of the default protection function.
=STD	The protection attributes of the new job variable are cataloged with the default system values.
ACCESS	Specifies either read-only or read/write access to a job variable.
=WRITE	Default value if STATE=NEW; read and write access to the job variable is permitted.
=READ	Only read access to the job variable is permitted.
SHARE	Specifies whether or not the job variable can be used by any other user ID.
=NO	Default value if STATE=NEW; the job variable must not be used by any other user IDs.
=YES	Specifies that the job variable may also be used by other user IDs.

OWNERAR	Specifies the access rights of the owner of the job variable (and of systems support). Read and write authorization must be explicitly assigned in each case. This operand not allowed in combination with VERSION=0.
=NO-ACCESS	The owner is explicitly denied both read and write authorization. This is equivalent to specifying (READ=NO,WRITE=NO).
=(READ=...,WRITE=...)	Read and write authorization are assigned as specified explicitly (READ/WRITE=YES) or not assigned (READ/WRITE=NO).
GROUPAR	Specifies the access rights for all user IDs from the same group as the owner (except for the owner and systems support). This operand not allowed in combination with VERSION=0. The definition of user groups is only possible if the software product SECOS is used. With a view to the possible use of SECOS, the same rights should be assigned to GROUP as to OTHERS.
=NO-ACCESS	User IDs of the owner group are explicitly denied both read and write authorization. This is equivalent to specifying (READ=NO, WRITE=NO).
=(READ=...,WRITE=...)	Read and write authorization are assigned as specified explicitly (READ/WRITE=YES) or not assigned (READ/WRITE=NO).
OTHERAR	Specifies the access rights of other users who do not belong to the owner group. This operand not allowed in combination with VERSION=0. If SECOS is not used, the access rights should nonetheless be set in the same way as for the owner group (GROUP) with a view to future possible use of SECOS.
=(READ=...,WRITE=...)	Read and write authorization are assigned as specified explicitly (READ/WRITE=YES) or not assigned (READ/WRITE=NO).

BASACL — Specifies whether a basic ACL is to be deleted, activated, or newly created for the job variable.
This operand not allowed in combination with VERSION=0.

=NONE — An activated basic ACL for the job variable is deleted. When a new job variable is created (STATE=NEW), NONE is the default value, meaning that no basic ACL is defined.Access control is then effected in accordance with the values ACCESS and SHARE (standard access control).

=STD — When the job variable is newly created (STATE=NEW), a basic ACL is created in which read and write access are permitted for the user group OWNER, and the user groups GROUP and OTHERS have no access rights.
If the job variable entry is updated (STATE=UPDATE), the value STD is evaluated only if no basic ACL was active. In this case a basic ACL is activated in which the access rights are set in accordance with the standard access control in the job variable entry (SHARE and ACCESS):

Standard access control		**BASIC-ACL protection**					
SHARE	**ACCESS**	**OWNER**		**GROUP**		**OTHERS**	
		R	**W**	**R**	**W**	**R**	**W**
NO	WRITE	Y	Y	N	N	N	N
NO	READ	Y	N	N	N	N	N
YES	WRITE	Y	Y	Y	Y	Y	Y
YES	READ	Y	N	Y	N	Y	N

Table 30: Standard access control/BASIC-ACL (CATJV macro)

GUARDS — Specifies whether a guard is to be deleted, activated or created for the job variable.
Specification of this operand is permissible only in conjunction with VERSION=3 or higher.

=NONE — A previously defined guard is cancelled. When a new job variable is created (STATE=NEW), NONE is the default value, meaning that no guard is defined.

=(READ=...,WRITE=...)
Read and/or write access is controlled via the specified guard (READ/WRITE=readguard/writeguard) or is not permitted at all (READ/WRITE=*NONE).
The name of the guard may be up to 8 characters long if specified without user ID. No catalog ID may be specified.

MANCLAS	*This operand is only evaluated for SM pubsets.* Specifies whether the HSMS functions JV backup, archival and (long term) archival are to be controlled via a management class defined via HSMS. See the manual “HSMS” [12] for further details. Specification of this operand is permissible only in conjunction with VERSION=3 or higher.
=NONE	The controlling of HMHS functions via a management class is terminated. When a job variable is renewed (STATE=NEW), NONE is the default value and means that no management class is defined.
=manclas	Name of the management class defined with HSMS.
RDPASS	Defines a read password for the job variable or cancels a previously defined read password.
=NONE	Cancels a previously defined read password. When a new job variable is created (STATE=NEW), NONE is the default and means that no read password is defined.
=password	Defines the read password that must be given in order to access the job variable. Length ≤ 4 bytes. It must be a C string, an X string or a decimal number. C string: C'character constant'; max. 4 characters. X string: X'hexadecimal constant'; max. 8 characters. Decimal number: -2147483648 ≤ number ≤ 2147483647 A password in the form X'00000000' or '0' is ignored. When a job variable is protected only by a read password, this password must also be specified in order to modify the job variable.
WRPASS	Defines a write password for the job variable or cancels a previously defined write password.
=NONE	Cancels a previously defined write password. When a new job variable is created (STATE=NEW), NONE is the default and means that no write password is defined.
=password	Defines the write password that must be given in order to have write access to the job variable. Length ≤ 4 bytes. It must be a C string, an X string or a decimal number (see the RDPASS operand).

RETPD	Specifies the retention period for the job variable.
=0	Default value if STATE=NEW; the retention period is zero days. This means that the expiration date field of the job variable entry contains today's date. (see the IDJEEXD field in the DSECT of the STAJV macro, page 180).
=days	Number of days that the job variable is to remain safe against modification even if the other protection attributes permit write access. The expiration date in the job variable entry contains the date on which protection is removed. The expiration date is the current date plus the specified number of days. This operand is ignored unless STATE=UPDATE is specified, i.e. the job variable must already be cataloged. The maximum value for RETPD is 32767 days.
MONJV	Specifies whether the protection attributes of a monitoring job variable are to remain unchanged.
=<u>UNCHANGED</u>	Is the default: the protection for a job-monitoring job variable is to remain unchanged.
=NO	Cancels protection of the system area (bytes 1 through 128) for a monitoring job variable. This specification is only effective in combination with STATE=UPDATE. Prior to the call, care must be taken to ensure that the monitored job has actually been removed from the queue (SHOW-JOB-STATUS).
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.
VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. The operand MF=D/C, which generates a DSECT or CSECT respectively, is not supported by this version (see note on DSECT).
=1	The macro expansion is compatible with JV V10.0.
=2	The macro expansion is compatible with JV V11.0 and V11.2.
=3	The macro expansion is compatible with JV V12.0.
=4	The macro expansion is compatible with JV ≥ V13.0C.

Notes concerning the DSECT

– Calling the macro with the operands MF=D and VERSION=1/2/3/4 generates a DSECT for the operand list of the CATJV macro of the corresponding version.

– A DSECT for the macro with VERSION=0 is generated by calling the macro IDJCA [D][,prefix].

Note on the assignment of access rights using a basic ACL:

A basic ACL is activated if basic ACL rights are specified for at least one authorized user (in the OWNERAR, GROUPAR or OTHERAR operand) When a basic ACL is activated, the rights for a user group that is not specified are set as though BASIC-ACL=NO-ACCESS (neither write nor read access rights) was specified for this user group.

Return information and error flags

see page 229

DSECT

```
  CATJV    CATJV MF=D,VERSION=4
1 **********************************************************************
1 *        VERSION 410
1 **********************************************************************
1 *       C A T J V   P A R A M E T E R   L I S T                      *
1 **********************************************************************
1          #INTF REFTYPE=REQUEST,                                        C
1                INTNAME=CATJV,INTCOMP=004
1 CATJV    DSECT
1 **********************************************************************
1 *       UNIT=41, FUNCTION=4,   VERSION=<PARAMETER VERSION>           *
1 **********************************************************************
1          FHDR   MF=(C,IDJC)
2          DS     0A
2 IDJCFHE  DS     0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJCIFID DS     0A              0   INTERFACE IDENTIFIER
2 IDJCFCTU DS     AL2             0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJCFCT  DS     AL1             2   FUNCTION NUMBER
2 IDJCFCTV DS     AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJCRET  DS     0A              4   GENERAL RETURN CODE
```

```
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJCSRET DS    0AL2             4   SUB RETURN CODE
2 IDJCSR2  DS    AL1              4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJCR2OK EQU   X'00'                All correct, no additional info
2 IDJCR2NA EQU   X'01'                Successful, no action was necessary
2 IDJCR2WA EQU   X'02'                Warning, particular situation
2 IDJCSR1  DS    AL1              5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJCRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJCRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJCRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJCRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJCRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJCRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJCRIER EQU   X'20'                INTERNAL ERROR
2 IDJCRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJCRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJCRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJCRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJCRWLR EQU   X'81'                     "     LONG       "
2 IDJCRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJCRTNA EQU   X'81'                SS TEMPORARILY NOT AVAILABLE
2 IDJCRDH  EQU   X'82'                SS IN DELETE / HOLD
2 *
2 IDJCMRET DS    0AL2             6   MAIN RETURN CODE
2 IDJCMR2  DS    AL1              6   MAIN RETURN CODE 2
2 IDJCMR1  DS    AL1              7   MAIN RETURN CODE 1
2 *
```

```
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJCRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJCFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 **********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL CATJV PARAMETER LIST      *
1 **********************************************************************
1 IDJCHDRI      EQU   X'00290404',4
1 IDJRDPAS      DS    CL4                          READ PASSWORD
1         DS    XL12                      RESERVED
1 IDJWRPAS      DS    CL4                          WRITE PASSWORD
1         DS    XL12                      RESERVED
1 IDJRETPD      DS    H                            RETENTION PERIOD
1 IDJCFLAG      DS    X                            FLAGS
1 IDJSTATE      EQU   X'80'                        7-7 1=UPDATE, 0=NEW (STATE)
1 IDJACCES      EQU   X'40'                        6-6 1=READ, 0=WRITE(ACCESS)
1 IDJSHARE      EQU   X'20'                        5-5 1=YES, 0=NO     (SHARE)
1 IDJRPN        EQU   X'10'                        4-4 1=RDPASS NULL,STATE=U
1 IDJWPN        EQU   X'08'                        3-3 1=WRPASS NULL,STATE=U
1 IDJRETPN      EQU   X'04'                        2-2 1=RETPD NULL, STATE=U /
1 *                                                 RETPD NOT NULL,STATE=NEW
1 IDJACCEN      EQU   X'02'                        1-1 1=ACCESS NULL,STATE=U
1 IDJSHARN      EQU   X'01'                        0-0 1=SHARE NULL, STATE=U
1 *
1 IDJCFLG1      DS    X                            FLAGS
1 IDJCENCR      EQU   X'80'                        7-7 0=YES, 1=NO
1 *                                               (ENCRYPTION)
1 IDJTYPE       EQU   X'00'                        6-6 NOT USED (DEL. V11.2)
1 IDJMONJV      EQU   X'20'                        5-5 1=MONJV=NO,STATE=UPDATE
1 IDJCP2        EQU   X'10'                        4-4 1=P2 CALLER,0=P1 CALLER
1 IDJCECT       EQU   X'08'                        3-3 1=SET BY CMD PROCESSING
1 IDJBACLN      EQU   X'04'                        2-2 1=SET BASIC-ACL = NONE
1 IDJBACLS      EQU   X'02'                        1-1 1=SET BASIC-ACL = STD
1 IDJNSTEX      EQU   X'01'                        0-0 1=SET NOSTEP=EXISTING
1 *
1 IDJJAR        DS    0X                           ACCESS RIGHTS
1 IDJOWNER      DS    X                            OWNER
1 IDJGROUP      DS    X                            GROUP
1 IDJOTHER      DS    X                            OTHERS
1 *
1 IDJJAUS       EQU   X'80'                        7-7 1=USER CLASS SPECIFIED
1 IDJJARS       EQU   X'40'                        6-6 1=READ   SPECIFIED
1 IDJJAWS       EQU   X'20'                        5-5 1=WRITE  SPECIFIED
1 IDJJARO       EQU   X'08'                        3-3 1=SET READ
1 IDJJAWO       EQU   X'04'                        2-2 1=SET WRITE
1 *
1 IDJCFLG2      DS    X                     FLAGS
```

```
1 IDJGRDN       EQU   X'80'            7-7  :S: GUARDS = *NONE
1 IDJPVSS       EQU   X'10'            4-4  :S: GUARD PUBSET SPECIFIED
1 IDJRDGS       EQU   X'08'            3-3  :S: READ GUARD SPECIFIED
1 IDJWRGS       EQU   X'04'            2-2  :S: WRITE GUARD SPECIFIED
1 IDJMANSP      EQU   X'02'            1-1  :S: MANCLAS SPECIFIED
1 *
1 IDJPRFLG      DS    X                PROTECT FLAG
1 IDJPRNSP      EQU   X'80'            7-7  :S: PROTECT NOT SPEC(DEFAULT)
1 IDJPRSTD      EQU   X'40'            6-6  :S: PROTECT STD SPECIFIED
1 IDJACCSP      EQU   X'08'            3-3  :S: ACCESS SPECIFIED
1 IDJSHASP      EQU   X'04'            2-2  :S: SHARE  SPECIFIED
1 IDJRDPSP      EQU   X'02'            1-1  :S: RDPASS SPECIFIED
1 IDJWRPSP      EQU   X'01'            0-0  :S: WRPASS SPECIFIED
1          DS    X                       RESERVED
1 *
1 IDJRDG        DS    CL18             READ GUARD
1 IDJWRG        DS    CL18             WRITE GUARD
1 IDJPUBS       DS    CL4              GUARD PUBSET
1 IDJMANCL      DS    CL8              MANCLAS
1 *
1 IDJJV1        DS    CL54                    JVNAME
1 IDJJV2        DS    CL54                    RENAME JVNAME
1          DS    CL54                    RESERVED
1          DS    A                       RESERVED
1          DS    CL16                    RESERVED
1 IDJCJVS       DS    A                       RESERVED
1 IDJPLLEN      EQU   *-CATJV                 LENGTH
1 ***********************************************************************
1          SPACE
```

COPJV
Copy job variable

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The COPJV macro copies the contents of a job variable (send JV) into another job variable (receive JV). If required, the protection properties of the send JV can also be copied along with the contents, apart from an existing MONJV or CJC protection.

Both permanent and temporary JVs can be copied. Read access must be allowed for the send JV and write access must be allowed for the receive JV. If the receive JV does not exist it is created, whereby when creating a permanent JV, the maximum number allowed in the user entry cannot be exceeded (see also CREATE-JV command).

Privileged functions:

The system administrator (TSOS privilege) is the co-owner of all job variables by default (can therefore also create and copy job variables under all user IDs).

This co-ownership can be restricted for permanent job variables.

Macro call format and operand description

```
Operation     Operands
-----------------------------------------------------------------------
COPJV         <jvname1>,<jvname2>

              ,SAME= ⎧ *NO  ⎫
                     ⎩ *YES ⎭

              ,WRITE= ⎧ *REPLACE ⎫
                      ⎩ *NEW     ⎭

                   ⎧ S                 ⎫
                   ⎪ L                 ⎪
                   ⎪ D                 ⎪
              ,MF= ⎨ C                 ⎬ ,PREFIX= ⎧ IDJ ⎫
                   ⎪ (E,(1))           ⎪          ⎩ pre ⎭
                   ⎪ (E, (<r>))        ⎪
                   ⎩ (E, <relexp>)     ⎭
```

<jvname1>	Name of the job variable to be copied (send JV). Read access must be allowed (with a JV under a foreign user ID, either USER-ACCESS=ALL-USERS must exist or read rights via BASIC-ACL or GUARDS, or co-ownership).
,<jvname2>	Name of the job variable into which copying is to be done (receive JV). If the receive JV is not cataloged, it is created. In this case, you can only specify your own ID or one for which you are co-owner. If the receive JV is cataloged, write access must be allowed (for a JV under a foreign user ID, either standard access control with USER-ACCESS=*ALL-USERS must exist or write rights via BASIC-ACL or GUARDS, or co-ownership). However, the receive JV will only be overwritten if REPLACE=*YES (default) is specified.
SAME	Specifies whether the protection properties of the send JV are also to apply for the receive JV.

=*NO	The protection properties are not taken over to the receive JV. If a receive JV is newly created, the system default values are set for the protection properties (see also the CREATE-JV command default settings). The previous protection properties remain intact for a receive JV that already exists.
=*YES	The receive JV is given the same protection properties as the send JV (regarding ACCESS, USER-ACCESS, OWNER, GROUP, OTHERS, EXPIR-DATE, EXPIR-TIME, MAN-CLASS, negotiated GUARDS, and the same passwords; see also the output fields of the SHOW-JV-ATTRIBUTES command). However, an existing MONJV or CJC protection is not taken over. Specifying PROTECTION=*SAME is ignored in the following cases (i.e. *NO applies): – The receive JV is a temporary JV. – The receive JV is being used by CJC. – The receive JV is under a foreign ID and the caller is not a co-owner. If the send JV is on a foreign user ID and protected with BASIC-ACL or GUARDS, the properties USER-ACCESS, BASIC-ACL and GUARDS of the receive JV are set to default values.
WRITE	Specifies whether an existing receive JV is to be overwritten.
=*REPLACE	An existing receive JV is overwritten without any message.
= *NEW	An existing receive JV is not overwritten. The command is rejected. The error handling is triggered in procedures (spin-off mechanism in non-S procedures or SDF-P error handling in S procedures).
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.

Return information and error flags

see page 229

DSECT

```
  COPJV    COPJV MF=D
1 ***********************************************************************
1 *        VERSION 410
1 ***********************************************************************
1 *       C O P J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                         C
1                INTNAME=COPJV,INTCOMP=001
1 COPJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=35,  VERSION=1                              *
1 ***********************************************************************
1          FHDR   MF=(C,IDJP)
2          DS    0A
2 IDJPFHE  DS    0XL8             0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJPIFID DS    0A               0   INTERFACE IDENTIFIER
2 IDJPFCTU DS    AL2              0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJPFCT  DS    AL1              2   FUNCTION NUMBER
2 IDJPFCTV DS    AL1              3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJPRET  DS    0A               4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJPSRET DS    0AL2             4   SUB RETURN CODE
2 IDJPSR2  DS    AL1              4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJPR2OK EQU   X'00'                All correct, no additional info
2 IDJPR2NA EQU   X'01'                Successful, no action was necessary
2 IDJPR2WA EQU   X'02'                Warning, particular situation
2 IDJPSR1  DS    AL1              5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'            FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'    PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'            INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'    NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'    WAIT AND RETRY
```

```
2 *
2 IDJPRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJPRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJPRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJPRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJPRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJPRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJPRIER EQU   X'20'                INTERNAL ERROR
2 IDJPRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJPRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJPRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJPRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJPRWLR EQU   X'81'                     "     LONG       "
2 IDJPRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJPRTNA EQU   X'81'                SS TEMPORARILY NOT AVAILABLE
2 IDJPRDH  EQU   X'82'                SS IN DELETE / HOLD
2 *
2 IDJPMRET DS    0AL2            6   MAIN RETURN CODE
2 IDJPMR2  DS    AL1             6   MAIN RETURN CODE 2
2 IDJPMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJPRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJPFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ************************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL COPJV PARAMETER LIST      *
1 ************************************************************************
1 IDJPHDRI      EQU   X'00292301',4
1 IDJPFLG       DS    XL1                      FLAG
1 IDJPSAME      EQU   X'80'                    7-7 1=SAME=YES  0=NO
1 IDJPWRIT      EQU   X'40'                    6-6 1=WRITE=NEW 0=REPLACE
1 *                                                0=DESCRIPTOR GIVEN
1 IDJPRES       DS    XL3                      RESERVED
1 IDJPJV1        DS    CL54                     JVNAME1
1 IDJPJV2        DS    CL54                     JVNAME2
1 IDJPJVS       DS    A                        RESERVED
1 IDJPPLLN      EQU   *-COPJV               LENGTH OF DSECT
1 ************************************************************************
1         SPACE
```

CSWJV
Check and set job variable

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **CSWJV** macro can be called at program level to set a job variable conditionally (see also the MODIFY-JV-CONDITIONALLY command).

A specified job variable is compared with the contents of a test field. If they are the same, the job variable is overwritten with the contents of another field; if they are not the same, the job variable value is transferred to the test field.

During processing by the macro, the job variable to be checked is protected against access from other jobs.

Macro call format and operand description

Operation	Operands
CSWJV	$\begin{Bmatrix}\texttt{jvid}\\ \texttt{(jvid [,start [,length])}\end{Bmatrix}$,area1, area2 [,PASS=password] [,VERSION=$\begin{Bmatrix}\underline{\texttt{0}}\\ \texttt{1}\end{Bmatrix}$] [,MF=$\begin{Bmatrix}\underline{\texttt{S}}\\ \texttt{C}\\ \texttt{(E,..)}\\ \texttt{D}\\ \texttt{L}\end{Bmatrix}$] [,PREFIX=$\begin{Bmatrix}\underline{\texttt{IDJ}}\\ \texttt{pre}\end{Bmatrix}$]

jvid	Identifies the job variable to be checked or set. jvid can be: jvname — A fully qualified path name of a permanent or temporary job variable *jvlink — A valid job variable link name
start	Specifies the position of the first byte to be checked or set within the job variable value, where 1 ≤ start ≤ 256. Default value: = 1.
length	Specifies the number of bytes of the job variable value that are to be checked or set. The sum of “start” and “length” must not exceed 257. If “length” is not specified, the values entered in the appropriate record length fields of “area1” and “area2” apply.
area1	Symbolic address of the test field in the user program.

```
Format    | 1111xxxx | comparison value                |
          <-4 bytes->
          <--1111=total length ----------------------->
```

The current value of the job variable “jvid” is compared with “comparison value”. If they do not match, “comparison value” is overwritten by the job variable value (return code 0456 or message JVS0456).

area2	Symbolic address of the fixed value for the job variable.

```
Format    | 1111xxxx | fixed value                     |
          <-4 bytes->
          <--1111=total length ----------------------->
```

If the job variable value of “jvid” matches the comparison value in “area1”, the job variable value is overwritten by the fixed value.

PASS=password	Defines the password to be input for write access to the job variable (see the **CATJV** macro). If the job variable is protected by a password and the PASS operand is not specified, the password must be made known to the system by means of the ADD-PASSWORD command (e.g. via the **CMD** macro) prior to the first call of **CSWJV**.

MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.
VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. The operand MF=D/C, which generates a DSECT or CSECT respectively, is not supported by this version (see note on DSECT).
=1	The macro expansion is compatible with JV ≥ V10.0.

Notes concerning the DSECT

- Calling the CSWJV macro with the operands MF=D and VERSION=1 generates a DSECT for the operand list of the CSWJV macro (VERSION=1).
- A DSECT for the macro with VERSION=0 is generated by calling the macro IDJCS [D][,prefix].

Return information and error flags

see page 229

DSECT

```
  CSWJV    CSWJV MF=D,VERSION=1
1 *********************************************************************
1 *        VERSION 203
1 *********************************************************************
1 *       C S W J V   P A R A M E T E R   L I S T                     *
1 *********************************************************************
1          #INTF REFTYPE=REQUEST,                                      C
1                INTNAME=CSWJV,INTCOMP=001
1 CSWJV    DSECT
1 *********************************************************************
1 *       UNIT=41, FUNCTION=5,   VERSION=<PARAMETER VERSION>          *
1 *********************************************************************
1          FHDR   MF=(C,IDJW)
2          DS     0A
2 IDJWFHE  DS     0XL8             0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJWIFID DS     0A               0   INTERFACE IDENTIFIER
2 IDJWFCTU DS     AL2              0   FUNCTION UNIT NUMBER
2 *                                    BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJWFCT  DS    AL1             2    FUNCTION NUMBER
2 IDJWFCTV DS    AL1             3    FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJWRET  DS    0A              4    GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJWSRET DS    0AL2            4    SUB RETURN CODE
2 IDJWSR2  DS    AL1             4    SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJWR2OK EQU   X'00'                All correct, no additional info
2 IDJWR2NA EQU   X'01'                Successful, no action was necessary
2 IDJWR2WA EQU   X'02'                Warning, particular situation
2 IDJWSR1  DS    AL1             5    SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJWRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJWRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJWRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJWRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJWRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJWRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJWRIER EQU   X'20'                INTERNAL ERROR
2 IDJWRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJWRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJWRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJWRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJWRWLR EQU   X'81'                     "     LONG       "
2 IDJWRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
```

```
2 IDJWRTNA EQU   X'81'              SS TEMPORARILY NOT AVAILABLE
2 IDJWRDH  EQU   X'82'              SS IN DELETE / HOLD
2 *
2 IDJWMRET DS    OAL2            6   MAIN RETURN CODE
2 IDJWMR2  DS    AL1             6   MAIN RETURN CODE 2
2 IDJWMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJWRLNK EQU   X'FFFF'            LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJWFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ***********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL CSWJV PARAMETER LIST      *
1 ***********************************************************************
1 IDJWHDRI      EQU   X'00290501',4
1 IDJWAREA      DS    A                        AREA 1 ADDRESS
1 IDJWARE2      DS    A                        AREA 2 ADDRESS
1 IDJWJV        DS    CL54                     JVNAME
1 IDJWPOS       DS    H                        SUBSTRING START POSITION
1 IDJWLEN       DS    H                        SUBSTRING LENGTH
1          DS    XL2                      RESERVED
1 IDJWPASS      DS    CL4                      PASSWORD
1          DS    XL12                     RESERVED
1 IDJWFLAG      DS    X                        FLAGS
1 IDJWENCR      EQU   X'80'                    7-7 0=YES, 1=NO
1 *                                           (ENCRYPTION)
1 IDJWFGNV      EQU   X'00'                    6-6 NOT USED (DEL. V12)
1 IDJWJVIX      EQU   X'20'                    5-5 0=NO,  1=YES
1 *                                           (JVID INDEXED (SUBSTRING) )
1 IDJWFEX       EQU   X'10'                    4-4 0=NO,  1=YES
1 *                                           (PASSWORD GIVEN)
1 IDJWP2        EQU   X'08'                    3-3 0=P1 CALLER,1=P2 CALLER
1          DS    XL3                      RESERVED
1 IDJWJVS       DS    A                        RESERVED
1 IDJWPLLN      EQU   *-CSWJV                LENGTH OF DSECT
1 ***********************************************************************
1          SPACE
```

DCLJV
Define job variable link name

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **DCLJV** macro assigns a link name to a job variable and generates a catalog entry for a job variable if one does not already exist.

Notes

- The assignments of job variable link names to job variable names is recorded in the JV-LINK table of the job.
- The JV-LINK table may contain only one entry per link name. A job variable can, however, be assigned to more than one link name.

Macro call format and operand description

Operation	Operands
DCLJV	jvname [,LINK=*jvlink] $\texttt{,VERSION=}\left\{\begin{array}{c}\underline{\texttt{0}}\\ \texttt{1}\end{array}\right\}\ \texttt{,MF=}\left\{\begin{array}{l}\underline{\texttt{S}}\\ \texttt{C}\\ \texttt{(E,..)}\\ \texttt{D}\\ \texttt{L}\end{array}\right\}\ \texttt{,PREFIX=}\left\{\begin{array}{c}\underline{\texttt{IDJ}}\\ \texttt{pre}\end{array}\right\}$

jvname	Fully qualified path name of a permanent or temporary job variable. If the job variable already exists, the user must have access authorization. If it is a new job variable, it is cataloged under the specified name (corresponds to the CATJV macro with default values). In this case, only systems support may specify a different user ID.
LINK	Specifies the link name (with * as the first character) by which the job variable can be addressed within the job from this point on. If an assignment for the specified link name already exists, the old link name assignment is replaced by the new assignment.
=*jvlink	Link name for the job variable. The link name may be up to 8 characters long, including the asterisk (*).
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.
VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. The MF=D/C operand, which generates a DSECT or CSECT respectively, is not supported by this version (see note on DSECT).
=1	The macro expansion is compatible with JV ≥ V10.0.

Notes concerning the DSECT

– Calling the DCLJV macro with the operands MF=D and VERSION=1 generates a DSECT for the operand list of the DCLJV macro (VERSION=1).
– A DSECT for the macro with VERSION=0 is generated by calling the macro IDJDC [D][,prefix].

Return information and error flags

see page 229

DSECT

```
  DCLJV    DCLJV MF=D,VERSION=1
1 **********************************************************************
1 *        VERSION 203
1 **********************************************************************
1 *       D C L J V   P A R A M E T E R   L I S T                      *
1 **********************************************************************
1          #INTF REFTYPE=REQUEST,                                       C
1                INTNAME=DCLJV,INTCOMP=001
1 DCLJV    DSECT
1 **********************************************************************
1 *       UNIT=41, FUNCTION=6,   VERSION=<PARAMETER VERSION>           *
1 **********************************************************************
1          FHDR   MF=(C,IDJD)
2          DS    0A
2 IDJDFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJDIFID DS    0A              0   INTERFACE IDENTIFIER
2 IDJDFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJDFCT  DS    AL1             2   FUNCTION NUMBER
2 IDJDFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJDRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJDSRET DS    0AL2            4   SUB RETURN CODE
2 IDJDSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJDR2OK EQU   X'00'               All correct, no additional info
2 IDJDR2NA EQU   X'01'               Successful, no action was necessary
2 IDJDR2WA EQU   X'02'               Warning, particular situation
2 IDJDSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
```

```
2 *
2 IDJDRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJDRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJDRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJDRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJDRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJDRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJDRIER EQU   X'20'                INTERNAL ERROR
2 IDJDRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJDRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJDRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJDRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJDRWLR EQU   X'81'                     "     LONG       "
2 IDJDRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJDRTNA EQU   X'81'                SS TEMPORARILY NOT AVAILABLE
2 IDJDRDH  EQU   X'82'                SS IN DELETE / HOLD
2 *
2 IDJDMRET DS    0AL2            6    MAIN RETURN CODE
2 IDJDMR2  DS    AL1             6    MAIN RETURN CODE 2
2 IDJDMR1  DS    AL1             7    MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJDRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJDFHL  EQU   8               8    GENERAL OPERAND LIST HEADER LENGTH
2 *
1 **********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL DCLJV PARAMETER LIST      *
1 **********************************************************************
1 IDJDHDRI      EQU   X'00290601',4
1 IDJDJVL       DS    CL8                      JV LINKNAME
1 IDJDJV        DS    CL54                     JVNAME
1 IDJDFLG       DS    XL2                      FLAG RESERVED
1 IDJDJVS       DS    A                        RESERVED
1 IDJDPLLN      EQU   *-DCLJV               LENGTH OF DSECT
1 **********************************************************************
1         SPACE
```

DONEVT
Delete condition for job variable event

General

Domain:	Job variables
Macro type:	Type S (with version=0): standard form/D/E/L form Type S (with version=1): standard form/C/D/E/L/M form, see section “The MF operand” on page 88

Macro description

The **DONEVT** macro is used to delete one or more conditions previously linked by means of an **ONEVT** macro to an event item declared with the **ENAEI** macro. The user can also delete all currently existing conditions for job variable events.

The user must issue the **DONEVT** macro before the **DISEI** macro, i.e. delete a condition before specifying the related event item. If this is not done, the results will be incorrect. Conditions for job variable events can be deleted in the following three ways:

– The ONEVT counter (COUNT operand) reaches the value zero.
– The program terminates.
– A DONEVT macro deletes the condition.

Macro call format and operand description

```
Operation   Operands
-----------------------------------------------------------------------------
DONEVT             ⎧ *ALL ⎫
              EIID=⎨ addr ⎬
                   ⎩ (r)  ⎭

              ⎧                   ⎧ S       ⎫                                   ⎫
              ⎪                   ⎪ (D,pre) ⎪           ⎧ DON ⎫                 ⎪
              ⎪ ,VERSION=0  ,MF= ⎨ D       ⎬ ,PREFIX= ⎨     ⎬                 ⎪
              ⎪                   ⎪ (E,...) ⎪           ⎩ pre ⎭                 ⎪
              ⎪                   ⎩ L       ⎭                                   ⎪
              ⎨                   ⎧ S ⎫                                         ⎬
              ⎪                   ⎪ C ⎪                                         ⎪
              ⎪ ,VERSION=1][,MF= ⎨ D ⎬  [,PARAM=⎧ addr ⎫],PREFIX=⎧ J   ⎫        ⎪
              ⎪                   ⎪ E ⎪          ⎩ (r)  ⎭          ⎩ pre ⎭        ⎪
              ⎪                   ⎪ L ⎪                                         ⎪
              ⎪                   ⎩ M ⎭   ,MACID=⎧ VSC   ⎫[,POST=⎧ val ⎫]        ⎪
              ⎩                                  ⎩ macid ⎭        ⎩ r   ⎭         ⎭
-----------------------------------------------------------------------------
```

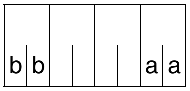
EIID	Specifies the conditions to be deleted which were previously set for a job variable event using the ONEVT macro.
=*ALL	All conditions which still exist for job variable events are deleted.
=addr	Symbolic address of the event item ID.
=(r)	Register containing the symbolic address of the event item ID.
MF PREFIX MACID PARAM	For a description of the MF, PREFIX, MACID and PARAM operands, see page 88. The permitted values and defaults for MF for this macro are indicated at the beginning of the macro description, those for PREFIX and MACID in the description of the VERSION operand.

VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. With the D form of this macro call a prefix (pre=1...3 letters) can be specified. Default: pre=DON
=1	The macro expansion is compatible with JV ≥ V10.0. With the C form, D form or M form of the macro call a prefix PREFIX (p=1 letter) can be specified, and with the C and D forms MACID also.
POST	Specifies a 2-byte value that can be used at program level to identify the DONEVT (in combination with the event item ID). The value must have been set with the ONEVT macro. Specification of this operand is permissible only in conjunction with VERSION=1 or higher.
=val	2-byte Assembler constant of any format.
=r	Register containing the address of the 2-byte field in which the user stored the value.

Return information and error flags

With VERSION=0:

R15



A return code relating to execution of the DONEVT macro (VERSION=0) is passed in register R15:
aa = return switch, bb = secondary indicator)

X'bb'	X'aa'	Meaning
X'00'	X'00'	Execution terminated normally
X'00'	X'04'	Function not executed: invalid address for event item
X'04'	X'04'	Function not executed: event item not found
X'08'	X'04'	Function not executed: system error (space management)
	X'FF'	Function not executed: CJC not available in the system

With VERSION=1:

Following initialization of the standard header (when MF=S/L is specified in the call), the return information is made available at the symbolic address <PREFIX><MACID>RET (4 bytes).

Standard-header

c	c	b	b	a	a	a	a

A return code relating to execution of the DONEVT macro (VERSION=1) is passed in the standard header:
(aaaa = main code, bb = subcode1, cc = subcode2)

X'cc'	X'bb'	X'aaaa'	Meaning
X'00'	X'00'	X'0000'	Execution terminated normally
X'00'	X'01'	X'0004'	Function not executed: invalid address for event item
X'04'	X'01'	X'0004'	Function not executed: event item not found
X'08'	X'20'	X'0004'	Function not executed: system error
		X'FFFF'	Function not executed: error during initialization of the standard header (see page 229)

In addition to the return codes named in VERSION=0 or 1, the general JV return codes for basic errors (e.g. invalid address for the operand list) may also appear. To find their meaning, see the relevant explanation under macro IDEJVS (page 229).

DSECT

```
  DONEVT   DONEVT MF=D,VERSION=1,PREFIX=A
1 DONEVT   MFCHK MF=D,                                                  C
1                SUPPORT=(C,D,E,L,M,S),                                 C
1                PREFIX=A,                                              C
1                MACID=VSC,                                             C
1                DMACID=VSC,                                            C
1                DNAME=DONPL,                                           C
1                PARAM=,                                                C
1                SVC=190
2 DONEVT   DSECT ,
2                *,##### PREFIX=A, MACID=VSC #####
1 AVSCFHDR   FHDR  MF=(C,AVSC),EQUATES=NO
2 AVSCFHDR DS    0A
2 AVSCFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 AVSCIFID DS    0A              0   INTERFACE IDENTIFIER
2 AVSCFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                 MUST BE RESET UNTIL FURTHER NOTICE
2 *                                 BIT 14-12 UNUSED, MUST BE RESET
2 *                                 BIT 11-0  REAL FUNCTION UNIT NUMBER
2 AVSCFCT  DS    AL1             2   FUNCTION NUMBER
2 AVSCFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 AVSCRET  DS    OA              4   GENERAL RETURN CODE
2 AVSCSRET DS    OAL2            4   SUB RETURN CODE
2 AVSCSR2  DS    AL1             4   SUB RETURN CODE 2
2 AVSCSR1  DS    AL1             5   SUB RETURN CODE 1
2 AVSCMRET DS    OAL2            6   MAIN RETURN CODE
2 AVSCMR2  DS    AL1             6   MAIN RETURN CODE 2
2 AVSCMR1  DS    AL1             7   MAIN RETURN CODE 1
2 AVSCFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 AVSCPCON   DS    X                        PARAM.-CONTROL
1 AVSCPREG   EQU   X'40'                    PARAM. IN REGISTER
1 AVSCXREG   EQU   X'20'                      POST IN REGISTER
1 AVSCXSPE   EQU   X'04'                      POST SPECIFIED
1 AVSCPTPR   EQU   X'02'                    P2 CALLER
1 AVSCPKEY   EQU   X'01'                    PARAM. IS KEYWORD
1 AVSCUNUD   DS    CL1                      UNUSED
1 AVSCPOST   DS    H                          POST-VALUE
1          ORG   AVSCPOST
1 AVSCPOSR   DS    AL1                        REG. CONT. POST-VALUE
1          DS    AL1                          NOT USED IN THIS CONTEXT
1 AVSCEIID    DS    A                       A(IDENTIFIER)
1          ORG   AVSCEIID
1 AVSCREG    DS    X                        REGISTER #
1 AVSCUNU2   DS    CL3                      UNUSED
1 AVSC#      EQU   *-AVSCFHDR               LENGTH
1 *
1 * RETURNCODES
1 *
1 AVSCOK     EQU   X'00000000'              SUCCESSFUL CALL
1 AVSCINAD   EQU   X'00010004'              INVALID PARM.-LIST
1 AVSCINEI   EQU   X'04010004'              INVALID IDENTIFIER
1 AVSCSYSE   EQU   X'08200004'              SYSTEM ERROR
```

ERAJV
Erase job variable

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **ERAJV** macro erases one or more job variable entries from the file catalog, or only the job variable value (in which case the job variable length is reset to zero).

The job variable name can be entered partially qualified and with wildcards.

Macro call format and operand description

Operation	Operands
ERAJV	jvid
	,IGNORE= { NONE / protect1 / (protect1,...4) }
	,CHECK= { STD / NO / MULTIPLE / PVS / SINGLE }
	,PASS= { NONE / password1 / (password1,...3) }

Operation	Operands
ERAJV (cont.)	$,\mathtt{DATA}=\left\{\begin{array}{c}\underline{\mathtt{NO}}\\ \mathtt{YES}\end{array}\right\}$ $,\mathtt{VERSION}=\left\{\begin{array}{c}\underline{\mathtt{0}}\\ \mathtt{1}\end{array}\right\}\ ,\mathtt{MF}=\left\{\begin{array}{l}\underline{\mathtt{S}}\\ \mathtt{C}\\ \mathtt{(E,..)}\\ \mathtt{D}\\ \mathtt{L}\end{array}\right\}\ ,\mathtt{PREFIX}=\left\{\begin{array}{c}\underline{\mathtt{IDJ}}\\ \mathtt{pre}\end{array}\right\}$

jvid

Identifies the job variable that is to be erased.
jvid can be:

jvname	A fully or partially qualified path name of a permanent or temporary job variable. The use of wildcards is allowed. Only systems support may specify a different user ID or enter a user ID with wildcard characters. The specified path name must consist of at least one of the three elements: catalog ID, user ID, JV name.
*jvlink	A valid job variable link name.
temp	All temporary job variables of the job are erased. “temp” stands for the special character defined with the system parameter TEMPFILE, used to identify temporary files and job variables (if necessary, ask systems support which character has been defined).

IGNORE

Specifies whether protection attributes included in the catalog for the job variable are to be ignored on erasure.
Non-privileged users can ignore the protection attributes ACCESS=READ, BASIC-ACL and EXDATE greater than the current date. The operand only takes effect when VERSION=1 is specified.

=<u>NONE</u>	Default value; causes an error message to be issued if an attempt is made to erase a job variable which possesses one of the specified protection attributes.
=(protect1,...4) =protect1	Specifies the protection attribute to be ignored. The following values can be specified individually or in a list:

ACCESS:	The protection attributes ACCESS=READ and BASIC-ACL are ignored.
EXDATE:	An expiration date greater than the current date is ignored.
RDPASS:	*Privileged users only* Password protection is ignored.
WRPASS:	*Privileged users only* Protection with a write password is ignored.

CHECK	Specifies whether it is possible to influence the erasure of job variables. Option for intervention: response to system query as to whether erasure is really to take place. In batch mode, this operand is dynamically adapted in the event of a conflict, as only NO is meaningful in batch mode. The operand only takes effect when VERSION=1 is specified
=<u>STD</u>	Default value; defines the following: MULTIPLE for interactive mode (global query) and NO for procedure/batch mode (erasure without prior warning).
=NO	The erasure of job variables cannot be additionally influenced.
=MULTIPLE	Results in a global query for each user ID (message: JVS0465), but only if more than one job variable is to be erased. MULTIPLE is the default value in interactive mode.
=PVS	Results in a global query for each pubset (message: JVS0468), but only if more than one job variable is to be erased.

=SINGLE A query is made for each job variable that is to be erased (message: JVS0469).

Note

With MULTIPLE, PVS and SINGLE, the following query is appended to each message:

```
REPLY (Y=YES; N=NO; T=TERMINATE; [,CHECK=mode]) )
```

Response	**Effect**
Y	Erase request is executed
N	Erase request is not executed, message JVS046A
T	Erasure is aborted
Any other response	Same effect as “N”
Optional suffix: CHECK=	Check mode is not modified.
CHECK=<mode>	Check mode (STD, NO, MULTIPLE, PVS or SINGLE) is set.

Table 31: Response options in the check dialog with ERAJV

PASS Enables the user to erase password-protected job variables. The operand only takes effect when VERSION=1 is specified. Passwords specified here are not entered in the password table of the job.

=<u>NONE</u> Default value; prevents the erasure of password-protected job variables.

=(password1,...3)
=password1 Specifies the password that is to be ignored when job variables are erased. Up to three passwords can be specified in a list.

DATA Determines the scope of erasure. The operand only takes effect when VERSION=1 is specified

=<u>NO</u> Default value; the job variable entry is erased.

=YES The value of the job variable is reset to zero.

MF
PREFIX For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.

VERSION	Specifies which version the macro expansion is to be compatible with.
=<u>0</u>	Default value: the macro expansion is compatible with JV ≤ V8.7. The operand MF=C/D, which generates a CSECT or DSECT, is not supported by this version (see note on DSECT).
=1	The macro expansion is compatible with JV ≥ V10.0.

Notes concerning the CSECT/DSECT

– Calling the ERAJV macro with the operands MF=D and VERSION=1 generates a DSECT for the operand list of the ERAJV macro (VERSION=1).
– A CSECT or DSECT can be created by calling the IDJER macro for a ERAJV macro with VERSION=0. A CSECT is generated with IDJER [,prefix], and a DSECT with IDJER D[,prefix].

Return information and error flags

see page 229

DSECT

```
  ERAJV    ERAJV MF=D,VERSION=1
1 ***********************************************************************
1 *        VERSION 400
1 ***********************************************************************
1 *       E R A J V   P A R A M E T E R   L I S T                     *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                        C
1                INTNAME=ERAJV,INTCOMP=001
1 ERAJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=3,   VERSION=<PARAMETER VERSION>          *
1 ***********************************************************************
1          FHDR  MF=(C,IDJE)
2          DS    0A
2 IDJEFHE  DS    0XL8             0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJEIFID DS    0A               0   INTERFACE IDENTIFIER
2 IDJEFCTU DS    AL2              0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJEFCT  DS    AL1              2   FUNCTION NUMBER
2 IDJEFCTV DS    AL1              3   FUNCTION INTERFACE VERSION NUMBER
```

```
2 *
2 IDJERET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJESRET DS    0AL2            4   SUB RETURN CODE
2 IDJESR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJER2OK EQU   X'00'               All correct, no additional info
2 IDJER2NA EQU   X'01'               Successful, no action was necessary
2 IDJER2WA EQU   X'02'               Warning, particular situation
2 IDJESR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJERFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 IDJERPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJERFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 IDJERFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 IDJERVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJERAER EQU   X'04'               ALIGNMENT ERROR
2 IDJERIER EQU   X'20'               INTERNAL ERROR
2 IDJERCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJERECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 IDJERECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJERWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 IDJERWLR EQU   X'81'                    "     LONG       "
2 IDJERWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJERTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 IDJERDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 IDJEMRET DS    0AL2            6   MAIN RETURN CODE
2 IDJEMR2  DS    AL1             6   MAIN RETURN CODE 2
```

```
2 IDJEMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJERLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJEFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ***********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL ERAJV PARAMETER LIST      *
1 ***********************************************************************
1 IDJEHDRI      EQU   X'00290301',4
1 IDJEJV        DS    CL80                    JVNAME
1          DS    CL40                    RESERVED
1 IDJEFLAG      DS    X                       FLAGS
1 IDJEDATA      EQU   X'80'                   7-7 DATA=YES
1 *                                          SET JV-VALUE TO NULLSTRING
1 IDJEECT       EQU   X'40'                   6-6 SET BY CMD PROCESSING
1 *                                          (NOT USED BY MACROCALLER)
1 IDJELIST      EQU   X'20'                   5-5 LIST=YES
1 *                                          (NOT USED BY MACROCALLER)
1 IDJENPWT      EQU   X'10'                   4-4 NO PASSWORD TEST
1 *                                          (PRIV CALLER ONLY)
1 IDJESEVR      EQU   X'08'                   3-3 SEVER PROCESSING
1 *                                          (ONLY USED BY SEVER-CMD)
1 IDJETERM      EQU   X'04'                   2-2 TERM. PROCESSING
1 *                                          (ONLY USED BY LOGOFF-CMD)
1 IDJEENCR      EQU   X'02'                   1-1 NO ENCRYPTION OF PASSWD
1 *                                          REQUIRED
1 IDJENEWI      EQU   X'01'                   0-0 NEW INTERFACE(V9.5 UP)
1 IDJECHK       DS    X                       FLAGS FOR CHECK INDICATOR
1 IDJECNO       EQU   X'80'                   7-7 CHECK=NO
1 *                                          ALL JVS ARE DELETED - NO
1 *                                          FEEDBACK TO CALLER
1 IDJECMUL      EQU   X'40'                   6-6 CHECK=MULTIPLE
1 *                                          IT IS ASKED IF ALL JVS OF
1 *                                          THE CURRENT USERID SHOULD
1 *                                          BE DELETED
1 IDJECPVS      EQU   X'20'                   5-5 CHECK=PVS
1 *                                          IT IS ASKED IF ALL JVS OF
1 *                                          THE CURRENT PUBSET SHOULD
1 *                                          BE DELETED
1 IDJECSIN      EQU   X'10'                   4-4 CHECK=SINGLE
1 *                                          IT IS ASKED FOR EACH JV IF
1 *                                          THE THE JV SHOULD BE DELETED
1 IDJECDEF      EQU   X'08'                   3-3 CHECK=STD
1 *                                          DEFAULTS ARE USED
1 *                                          DIALOG: MULTIPLE
1 *                                          OTHERS: NO
```

```
1 IDJEIGNO      DS    X                         FLAGS FOR IGNORE PARAMETER
1 IDJEINON      EQU   X'80'                     7-7 IGNORE=NONE
1 *                                            ALL PROTECTIONS ARE CHECKED
1 *                                            IF PROTECTION DEFINED ERASE
1 *                                            IS REJECTED
1 IDJEIRDP      EQU   X'40'                     6-6 IGNORE=RDPASS
1 *                                            RDPASS PROTECTION IS
1 *                                            IGNORED. JV IS DELETED
1 IDJEIWRP      EQU   X'20'                     5-5 IGNORE=WRPASS
1 *                                            WRPASS PROTECTION IS
1 *                                            IGNORED. JV IS DELETED
1 IDJEIACC      EQU   X'10'                     4-4 IGNORE=ACCESS
1 *                                            ACCESS=READ IS IGNORED.
1 *                                            JV IS DELETED
1 IDJEIEXD      EQU   X'08'                     3-3 IGNORE=EXDATE
1 *                                            IGNORE RETPD DEFINITION.
1 IDJEFLG1      DS    X                         FLAG 1
1 IDJEP2        EQU   X'80'                     7-7 CALLER=P2
1 IDJNSTEP      DS    6XL1                      3 ERR'S
1 IDJERESE      DS    14XL1                     UNUSED
1 IDJEPWD1      DS    CL4                       PASSWORD1 SET DEFAULT 0
1 IDJEPWD2      DS    CL4                       PASSWORD2 SET DEFAULT 0
1 IDJEPWD3      DS    CL4                       PASSWORD3 SET DEFAULT 0
1 * THE FOLLOWING FIELD IS SUPPLIED ONLY FOR CMD PROCESSING
1 IDJEADDR      DS    CL4                       SORT TABLE ADDRESS
1 IDJEJVS       DS    A                         RESERVED
1 IDJEPLLN      EQU   *-ERAJV                   LENGTH OF DSECT
1 ***********************************************************************
1         SPACE
```

GETJV
Get job variable value

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **GETJV** macro transfers the value of a user or special job variable to an area of the user program.

Macro call format and operand description

Operation	Operands

GETJV	{ jvid | (jvid [,start [,length]]) },area,size
	
	[,PASS=password]
	
	,PARMOD={ <u>24</u> | 31 }
	
	,VERSION={ <u>0</u> | 1 } ,MF={ <u>S</u> | C | (E,..) | D | L } ,PREFIX={ <u>IDJ</u> | pre }

jvid	Identifies the job variable. “jvid” can be:

jvname	Fully qualified path name of a permanent or temporary job variable.
*jvlink	Valid job variable link name.

start	Start position for output.
length	Output length.
area	Address of an area in the user program to which the value of the job variable is to be transferred. The area is supplied as follows:

```
Format     | 1111xxxx | JV value                   |
           <-4 bytes->
           <--1111=total length ----------------------->
```

size	Specifies the size of “area”. It must be at least as long as the “length of the value to be read + 4” and may not be longer than 32767. The actual size of “area”, i.e. the length of the JV value +4, is entered in the first two bytes of the area. If the total length of the job variable value is greater than the maximum value “size - 4”, the job variable value is truncated so that the maximum value is not exceeded.
PASS=password	Read password.
PARMOD	Controls macro expansion. Either the 24-bit or the 31-bit interface is generated. PARMOD is evaluated *only when VERSION=0* applies. If PARMOD is not specified here, macro expansion is performed according to the specification for the GPARMOD macro or according to the default setting for the assembler (=24-bit interface).
=<u>24</u>	The 24-bit interface is generated. Data lists and instructions use 24-bit addresses. (Address space ≤ 16 Mb.)
=31	The 31-bit interface is generated. Data lists and instructions use 31-bit addresses. (Address space ≤ 2 Gb.) Data lists start with the standard header.
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.

VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. The operand MF=D/C, which generates a DSECT or CSECT respectively, is not supported by this version (see note on DSECT).
=1	The macro expansion is compatible with JV ≥ V10.0.

Note

In contrast to the command level, the length 0 may be specified in the parameter list, representing the total job variable length.

Notes concerning the DSECT

- Calling the GETJV macro with the operands MF=D and VERSION=1 generates a DSECT for the operand list of the GETJV macro (VERSION=1).
- A CSECT/DSECT for the macro with VERSION=0 is generated by calling the macro IDJGE [D][,prefix] [,PARMOD=24/31].

Return information and error flags

see page 229

DSECT

```
  GETJV    GETJV MF=D,VERSION=1
1 ***********************************************************************
1 *        VERSION 203
1 ***********************************************************************
1 *       G E T J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                         C
1                INTNAME=GETJV,INTCOMP=002
1 GETJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=0,   VERSION=1 (V9.0)                       *
1 *                              VERSION=2 (V10.0)                      *
1 ***********************************************************************
1          FHDR   MF=(C,IDJG)
2          DS     0A
2 IDJGFHE  DS     0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJGIFID DS     0A              0   INTERFACE IDENTIFIER
2 IDJGFCTU DS     AL2             0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJGFCT  DS    AL1             2   FUNCTION NUMBER
2 IDJGFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJGRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJGSRET DS    0AL2            4   SUB RETURN CODE
2 IDJGSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJGR2OK EQU   X'00'               All correct, no additional info
2 IDJGR2NA EQU   X'01'               Successful, no action was necessary
2 IDJGR2WA EQU   X'02'               Warning, particular situation
2 IDJGSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJGRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 IDJGRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJGRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 IDJGRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 IDJGRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJGRAER EQU   X'04'               ALIGNMENT ERROR
2 IDJGRIER EQU   X'20'               INTERNAL ERROR
2 IDJGRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJGRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 IDJGRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJGRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 IDJGRWLR EQU   X'81'                    "     LONG       "
2 IDJGRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
```

```
2 IDJGRTNA EQU   X'81'              SS TEMPORARILY NOT AVAILABLE
2 IDJGRDH  EQU   X'82'              SS IN DELETE / HOLD
2 *
2 IDJGMRET DS    0AL2           6   MAIN RETURN CODE
2 IDJGMR2  DS    AL1            6   MAIN RETURN CODE 2
2 IDJGMR1  DS    AL1            7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJGRLNK EQU   X'FFFF'            LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJGFHL  EQU   8              8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ***********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL GETJV PARAMETER LIST      *
1 ***********************************************************************
1 IDJGHDRI      EQU   X'00290002',4
1 IDJGAR31      DS    A                        AREA ADDRESS(31 BIT FORMAT)
1 IDJGSIZE      DS    H                        AREA SIZE
1 IDJGJV        DS    CL54                     JVNAME
1 IDJGPOS       DS    H                        SUBSTRING START POSITION
1 IDJGLEN       DS    H                        SUBSTRING LENGTH
1 IDJGPASS      DS    CL4                      PASSWORD
1          DS    XL12
1 IDJGFLAG      DS    X                        FLAGS
1 IDJGENCR      EQU   X'80'                    7-7 0=YES, 1=NO
1 *                                           (ENCRYPTION)
1 IDJGFGNV      EQU   X'40'                    6-6 0=NO,  1=YES
1 *                                           (NUMERIC-VALUE)
1 IDJGJVIX      EQU   X'20'                    5-5 0=NO,  1=YES
1 *                                           (JVID INDEXED (SUBSTRING) )
1 IDJGP2        EQU   X'10'                    4-4 0=P1 CALLER,1=P2 CALLER
1 IDJGECT       EQU   X'08'                    3-3 1=SET BY CMD PROCESSING
1 IDJGFGBV      EQU   X'04'                    2-2 0=NO,  1=YES
1 *                                           (BOOLEAN-VALUE)
1 IDJGNSTR      EQU   X'02'                    1-1 0=NO,  1=YES
1 *                                           (NULLSTRING DEFINED)
1          DS    XL3
1 IDJGJVS       DS    A                        RESERVED
1 IDJGPLLN      EQU   *-GETJV                LENGTH OF DSECT
1 ***********************************************************************
1          SPACE
```

JVSEL
Limiting the JV selection of the STAJV macros to specific attributes

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=JSE

Macro description

The JVSEL macro limits the job variable set which is transferred to the user area with a STAJV call, to job variables with specific attributes.

Macro call format and operand description

Operation	Operands
JVSEL	
	,ACCESS= $\left\{\begin{array}{l}\underline{\text{*ANY}}\\ \text{*READ}\\ \text{*WRITE}\end{array}\right\}$
	,SHARE= $\left\{\begin{array}{l}\underline{\text{*ANY}}\\ \text{*YES}\\ \text{*NO}\end{array}\right\}$
	,PASS= $\left\{\begin{array}{l}\underline{\text{*ANY}}\\ \text{*NONE}\\ \text{*RDPASS}\\ \text{*WRPASS}\\ \text{(list-of-pass)}\end{array}\right\}$

```
Operation   Operands

JVSEL
(cont.)
                         ⎧ *ANY                              ⎫
                         ⎪ *NONE                             ⎪
                         ⎪ date                              ⎪
                         ⎪ date(time[,])                     ⎪
                         ⎪ date(time1,time2)                 ⎪
                         ⎪ (date[,])                         ⎪
             [,CRDATE=   ⎨ (date(time)[,])                   ⎬]
                         ⎪ (,date)                           ⎪
                         ⎪ (,date(time))                     ⎪
                         ⎪ (date1,date2)                     ⎪
                         ⎪ (date1(time),date2)               ⎪
                         ⎩ (date1,(time),date2(time))        ⎭

                       ⎧ *ANY                              ⎫
                       ⎪ *NONE                             ⎪
                       ⎪ date                              ⎪
                       ⎪ date(time[,])                     ⎪
                       ⎪ date(time1,time2)                 ⎪
                       ⎪ (date[,])                         ⎪
             ,EXDATE=  ⎨ (date(time)[,])                   ⎬
                       ⎪ (,date)                           ⎪
                       ⎪ (,date(time))                     ⎪
                       ⎪ (date1,date2)                     ⎪
                       ⎪ (date1(time),date2)               ⎪
                       ⎩ (date1,(time),date2(time))        ⎭

                       ⎧ *ANY   ⎫
             ,BASACL=  ⎨ *NONE  ⎬
                       ⎩ *YES   ⎭

                         ⎧ *ANY         ⎫
             ,OWNERAR=   ⎨ *NO-ACCESS   ⎬
                         ⎩ access-list  ⎭
```

Operation	Operands
JVSEL (cont.)	,GROUPAR= *ANY *NO-ACCESS access-list
	,OTHERAR= *ANY *NO-ACCESS access-list
	,GUARDS= *ANY *NONE *YES (READ= *ANY / *NONE / fname ,WRITE= *ANY / *NONE / fname)
	,MANCLAS= *ANY *NONE <c-string 1..8>
	,MONJV= *ANY *YES *NO
	,CJC= *ANY *YES *NO
	,PROTACT= *ANY *LEVEL-0 *LEVEL-1 *LEVEL-2 (list-of-protact)

Operation	Operands

JVSEL
(cont.)
	,SIZE={ <u>*ANY</u> | nmbr | (nmbr[,]) | (,nmbr) | (nmbr1,nmbr2) }
	
	,TIMBASE={ <u>*LTI</u> | *UTC }
	
	,VERSION=<u>1</u> ,MF={ D | C | L } ,PREFIX={ <u>JSE</u> | pre }

ACCESS
Selects job variablesjob variableson the basis of the access mode

= <u>*ANY</u>
Default value; the access mode is not a selection criterion.

= *READ
Provides information about job variables for which only read access is permitted.

= *WRITE
Provides information about job variables for which only write access is permitted.

SHARE
Selects and retrieves information on job variables based on whether or not they are shareable. If a foreign user ID is specified with “$userid.”, SHARE=YES applies implicitly.

= <u>*ANY</u>
Default value; shareability is not used as a selection criterion.

= *YES
Returns information on all job variables that are shareable, i.e. which are also accessible to other user IDs under active standard access control.

= *NO
Returns information on all job variables that are not shareable, i.e. that are only accessible to the owner or co-owner under active standard access control.

PASS
Selects and returns information on job variables on the basis of the password protection defined with CATAL. If several password types are specified in the form of a list, all job variables that satisfy one of the named conditions (logical OR) are selected by the system. Passwords are not output.

= <u>*ANY</u>	Default value; password protection is not used as a selection criterion.
= *NONE	Selects job variables for which no password protection exists.
= *RDPASS	Selects job variables which are protected by means of a read password.
= *WRPASS	Selects job variables which are protected by a write password.
= (list-of-pass)	The user may specify more than one type of password in the form of a list. All job variables protected by one of the specified password types will be selected.

CRDATE

Returns information on job variables on the basis of their creation date (CRDATE); range specifications are inclusive of the limit values. It is meaningless to specify a CREATION-DATE which lies in the future!
Here, the user can specify the expiration date in either of two ways:

1. as an *absolute date value*
   a specific date in the form yymmdd or [yy]yy-mm-dd
   (yy = year, mm = month, dd = day)
2. as a *relative date value*
   (6 digits with preceding sign), the number of days from today's date, in the form -n for dates in the past, and [+]n for dates in the future; (yesterday ≙ -1 or today ≙ ±0)

= <u>*ANY</u>	Default value; the creation date is not a selection criterion.
= date	Returns information on all job variables that were created on the specified date.
= (date[,])	Returns information on all job variables that were created on or after the specified date (creation date ≥ current date).
= (,date)	Returns information on all job variables that were created on or before the specified date (creation date ≤ current date).
= (date1,date2)	Returns information on all job variables that were created within the specified period (date1 ≤ creation date ≤ date2).
= date(time[,])	Returns information on all job variables that were created on the specified date on or after the specified time.
= date(time1,time2)	Returns information on all job variables that were created on the specified date within the specified period.

= (date(time)[,])	Returns information on all job variables that were created on or after the specified date and time.
= (,date(time))	Returns information on all job variables that were created before the specified date and time.
= (date1(time),date2(time))	Returns information on all job variables that were created within the specified period. The upper and lower limits of the specified period are defined more precisely by a time specification in both cases.
EXDATE	Returns information on job variables on the basis of their "expiration date", i.e. the date from which write accesses to the job variable are allowed. It is meaningful to specify a date in the future if retention periods are being queried. Here, the user can specify the expiration date as an *absolute date value* or as a *relative date value* (see "CRDATE" on page 140).
= *ANY	Default value; the expiration date is not a selection criterion.
= date	Returns information on all job variables for which the specified expiration date is defined.
= (date[,])	Returns information on all job variables for which the expiration date is greater than or equal to the specified date.
= (,date)	Returns information on all job variables for which the expiration date is less than or equal to the specified date.
= (date1,date2)	Returns information on all job variables for which the expiration date lies within the specified period (date1 $\leq$ expiration date $\leq$ date2).
= date(time[,])	Returns information on all job variables with the specified expiration date and with an expiration time that is greater than or equal to the specified time. Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= date(time1,time2)
Returns information on all job variables with the specified expiration date and with a time of expiration that lies within the specified time interval.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date(time)[,])
Returns information on all job variables with the specified expiration date and with an expiration time that is greater than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (,date(time))
Returns information on all job variables for which the expiration date and time is less than or equal to the specified time.
Note that the time of expiration (i.e. the time on the expiration date) is always entered as 00:00:00 hours in the catalog at present!

= (date1(time),date2(time))
Returns information on all job variables for which the expiration date lies within the specified period (date1 ≤ expiration date ≤ date2). The upper and lower limits of the specified period are defined more precisely by time values in both cases.

BASACL	Returns information on files which are selected on the basis of a defined BASIC-ACL.
= <u>*ANY</u>	Default value; the BASIC-ACL is not a selection criterion.
= *NONE	Returns information on all job variables for which no BASIC-ACL entry is defined.
= *YES	Returns information on all job variables for which a BASIC-ACL entry is defined.
OWNERAR	Returns information on job variables selected on the basis of the access rights that are defined for the job variable owner in the BASIC-ACL entries.
= <u>*ANY</u>	Default value; BASIC-ACL entries for the job variable owner are not used as a selection criterion.
= *NO-ACCESS	Returns information on all job variables that the owner is not allowed to access.

= access-list Returns information on all job variables for which at least one of the listed access rights has been defined for the file owner in the BASIC-ACL entry.

access-list has the following format:

```
  ⎧ READ =  *YES ⎫     ⎧ WRITE =  *YES ⎫
  ⎪ R    =  *Y   ⎪     ⎪ W     =  *Y   ⎪
 [⎨ READ =  *NO  ⎬][,  ⎨ WRITE =  *NO  ⎬])
  ⎩ R    =  *N   ⎭     ⎩ W     =  *N   ⎭
```

The parentheses form part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=*YES or R=*Y	Selects all job variables that may be accessed by the owner for reading.
READ=*NO or R=*N	Selects all job variables that cannot be accessed by the owner for reading.
WRITE=*YES or W=*Y	Selects all job variables that can be accessed by the owner for writing.
WRITE=*NO or W=*N	Selects all job variables that cannot be accessed by the owner for writing.

GROUPAR Selects and returns information on job variables on the basis of the access rights that are defined for members of the job variable owner's user group in BASIC-ACL entries.

= <u>*ANY</u> The BASIC-ACL entries for members of the job variable owner's user group are not used as a selection criterion.

= *NO-ACCESS Returns information on all job variables that cannot be accessed by the user group of the owner.

= access-list Returns information on all job variables for which at least one of the access rights specified in the list has been entered for the user group of the job variable owner in the BASIC-ACL entry.

access-list has the following format:

```
  ⎧ READ =  *YES ⎫     ⎧ WRITE =  *YES ⎫
  ⎪ R    =  *Y   ⎪     ⎪ W     =  *Y   ⎪
 [⎨ READ =  *NO  ⎬][,  ⎨ WRITE =  *NO  ⎬])
  ⎩ R    =  *N   ⎭     ⎩ W     =  *N   ⎭
```

The parentheses form part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=*YES or R=*Y	Selects all job variables that may be accessed for reading by the user group of the owner.
READ=*NO or R=*N	Selects all job variables that cannot be accessed for reading by the user group of the owner.
WRITE=*YES or W=*Y	Selects all job variables that may be accessed for writing by the user group of the owner.
WRITE=*NO or W=*N	Selects all job variables that cannot be accessed for writing by the user group of the owner.

OTHERAR Selects and returns information on job variables based on the access rights that are defined via BASIC-ACL entries for all users other than the job variable owners's user group.

= *ANY Default value; the BASIC-ACL entries for all users other than the job variable owner's user group do not serve as a selection criterion.

= *NO-ACCESS Returns information on all job variables that may be accessed by users not belonging to the job variable owner's user group.

= access-list Returns information on all job variables for which at least one of the listed access rights has been defined for users not in the job variable owner's user group in the BASIC-ACL entries.

access-list has the following format:

```
  ⎧ READ =  *YES ⎫    ⎧ WRITE =  *YES ⎫
  ⎪ R    =  *Y   ⎪    ⎪ W     =  *Y   ⎪
[ ⎨ READ =  *NO  ⎬][, ⎨ WRITE =  *NO  ⎬])
  ⎩ R    =  *N   ⎭    ⎩ W     =  *N   ⎭
```

The parentheses form part of the operand value and must be specified.

The individual elements of the access list mean the following:

READ=*YES or R=*Y	Selects all job variables that can be accessed for reading by users who are not in the owner's user group.
READ=*NO or R=*N	Selects all job variables that cannot be accessed for reading by users who are not in the owner's user group.
WRITE=*YES or W=*Y	Selects all job variables that can be accessed for writing by users who are not in the owner's user group.
WRITE=*NO or W=*N	Selects all job variables that cannot be accessed for writing by users who are not in the owner's user group.

GUARDS	The user can select job variables to be processed on the basis of the access protection defined by GUARDS (see the “SECOS” [10] manual).
= *ANY	Default value; the access protection defined by GUARDS is not a selection criterion.
= *NONE	Returns information on all job variables which have no access protection defined by GUARDS.
= *YES	Returns information on all job variables which have access protection defined by GUARDS.
= (READ=...,WRITE=...)	The type of access protection provided by GUARDS that is to be used as a selection criterion can be defined by the user in a list. For each access mode (read and write), the defined protection can be specified precisely. If no entry is made for an access mode, the protection defined for that access mode has no effect on the selection.

For each access mode, one of the following values may be specified:

*ANY	The defined GUARDS protection is not a selection criterion.
*NONE	No guard has been defined for the specified access mode, i.e. the corresponding access is denied.
fname	All conditions for granting access in the specified access mode are defined in the guard fname.

MANCLAS	Returns in formation on all job variables according to the HSMS management class for file backup to SM pubsets.
= *ANY	Default value; the HSMS management class is not a selection criterion.
= *NONE	All files for which no HSMS management class is defined are selected.
= <c-string 1..8>	All files with the specified HSMS management class are selected.

MONJV	Selects job variables that are used as job monitoring job variables.
= *ANY	Preset value: use as a monitoring job variable is not a selection criterion.
= *NO	Selects job variables that do not monitor any jobs.

= *YES	Selects job variables that monitor jobs (also see the SHOW-JV-ATTRIBUTES command, the JV-TYPE IS MONJV output field in the “Commands” manual [1]]).

CJC	Selects job variables according to their use in CJC functions.
= <u>*ANY</u>	Default value; use as a job monitoring job variable is not a selection criteria.
= *NO	Selects job variables that are not used in CJC functions.
= *YES	Selects job variables that are used in CJC functions.

PROTACT	Selects and retrieves information on job variables on the basis of the protection level provided by the highest activated access control.

When the file is accessed, the highest activated protection level applies. The following table shows the method used for access control, the protection attribute to be specified in the CATJV macro and the file protection hierarchy (protection levels):

Access control method	**Protection attribute (CATJV macro operand)**	**Protection level**
Standard access control	ACCESS u. SHARE	0
Basic access control list	BASACL, OWNERAR, GROUPAR, OTHERAR	1
Access control using GUARDS	GUARDS	2

All other protection attributes of the file (e.g. passwords) are evaluated independently, without regard to the implemented protection level.

= <u>*ANY</u>	Default value; is the default value; returns information on all job variables without regard to the protection level of the highest activated access control.
= *LEVEL-0	Returns information on job variables for which access is controlled via standard access control.
= *LEVEL-1	Returns information on job variables for which access is controlled via a basic access control list (BASIC-ACL protection).
= *LEVEL-2	Returns information on job variables for which access is controlled via an access control list by GUARDS.
= (list-of-protact)	The user may specify up to a maximum of three protection levels in a list. All job variables for which the protection level of the access control method matches one of those specified are selected.

SIZE	Selects job variables according to the length of the job variable values. Range specifications include their respective limit values.
= <u>*ANY</u>	Default value; the length of the job variable value is not a selection criteria.
= nmbr	Selects job variables which are the specified number of bytes long. Possible values: $0 \leq nmbr \leq 256$
= (nmbr[,])	Returns information about job variables which are at least the specified number of bytes long (SIZE ≥ specified value).
= (,nmbr)	Returns information about job variables which have a length which is less than or equal to the specified number of bytes (SIZE ≤ specified value).
= (nmbr1,nmbr2)	Returns information about job variables which have a length within the specified range (nmbr1 ≤ length ≤ nmbr2, where nmbr1 < nmbr2).
TIMBASE	Specifies the timebase which is used when inputing creation or expiration dates.
= <u>*LTI</u>	Default value: date or time inputs are interpreted as inputs in local time.
= *UTC	Date or time inputs are interpreted as GMT.
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.
VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>1</u>	Default value: The macro expansion is compatible with JV V14.0.

Return information and error flags

Currently the only option of using a JVSEL parameter list is to use it as SELADDR in a STAJV. The latter supplies a return code in accordance with page 229 and, specifically in the case of STAJV return code 0491 (error during selection), it also records which selection criterion is faulty in the JSERC field of the JVSEL parameter list.

DSECT

```
  JVSEL    JVSEL MF=D
1 JVSEL    MFCHK MF=D,SUPPORT=(D,C),PREFIX=J,MACID=SE,DMACID=JSE
2 JVSEL    DSECT ,
2                *,##### PREFIX=J, MACID=JSE #####
1 *
1 *
1          DS    0A
1 JSETEXT  DC    C'SELECT'
1 JSEVERS  DC    X'01'
1 JSERC    DC    X'FF'
1 JSERCOK  EQU   X'00'                NO ERROR
1 JSERCCRD EQU   X'01'                INVALID CRDATE
1 JSERCEXD EQU   X'02'                INVALID EXDATE
1 JSERCCRT EQU   X'03'                INVALID CRTIME
1 JSERCEXT EQU   X'04'                INVALID EXTIME
1 JSERCUND EQU   X'FF'                UNDEFINED
1 *
1 JSESELI0 DC    B'00000000'       SELECT IND0
1 JSESICRD EQU   X'80'                CREATION-DATE
1 JSESIEXD EQU   X'40'                EXPIRATION-DATE
1 JSESIPWP EQU   X'20'                PASSWORD-PROTECTED
1 JSESIPRO EQU   X'10'                PROTECTION-LEVEL
1 JSESIACC EQU   X'08'                ACCESS
1 JSESISHR EQU   X'04'                SHARE
1 JSESIBAC EQU   X'02'                BASIC-ACL
1 JSESIGUA EQU   X'01'                GUARDS
1 *
1 JSESELI1 DC    B'00000000'       SELECT IND1
1 JSESISIZ EQU   X'80'                SIZE
1 JSESIMON EQU   X'40'                MONJV
1 JSESIMCL EQU   X'20'                MANAGEMENT-CLASS
1 JSESICJC EQU   X'10'                CJC
1 JSESI5UU EQU   X'0F'             -- UNUSED, MUST BE 0 --
1 *
1 JSEUNUS1 DC    X'0000'
1 *
1 JSEBACL  DC    B'00000000'       BASIC-ACL
1 JSEBACLY EQU   X'80'                YES
```

```
1 JSEBACLN EQU   X'40'              NONE
1 JSEBACLU EQU   X'2F'           -- UNUSED, MUST BE 0 --
1 *
1 JSEBOW   DC    B'00000000'     BASIC-ACL-OWNER
1 JSEBOWRS EQU   X'80'              READ-RIGHT-SPECIFIED
1 JSEBOWWS EQU   X'40'              WRITE-RIGHT-SPECIFIED
1 JSEBOWUN EQU   X'20'           -- UNUSED, MUST BE 0 --
1 JSEBOWRY EQU   X'10'              READ = YES
1 JSEBOWWY EQU   X'08'              WRITE = YES
1 JSEBOWNU EQU   X'04'           -- UNUSED, MUST BE 0 --
1 JSEBOWNO EQU   X'02'              NO-ACCESS
1 JSEBOWUU EQU   X'01'           -- UNUSED, MUST BE 0 --
1 *
1 JSEBGR   DC    B'00000000'     BASIC-ACL-GROUP
1 JSEBGRRS EQU   X'80'              READ-RIGHT-SPECIFIED
1 JSEBGRWS EQU   X'40'              WRITE-RIGHT-SPECIFIED
1 JSEBGRUN EQU   X'20'           -- UNUSED, MUST BE 0 --
1 JSEBGRRY EQU   X'10'              READ = YES
1 JSEBGRWY EQU   X'08'              WRITE = YES
1 JSEBGRNU EQU   X'04'           -- UNUSED, MUST BE 0 --
1 JSEBGRNO EQU   X'02'              NO-ACCESS
1 JSEBGRUU EQU   X'01'           -- UNUSED, MUST BE 0 --
1 *
1 JSEBOT   DC    B'00000000'     BASIC-ACL-OTHERS
1 JSEBOTRS EQU   X'80'              READ-RIGHT-SPECIFIED
1 JSEBOTWS EQU   X'40'              WRITE-RIGHT-SPECIFIED
1 JSEBOTUN EQU   X'20'           -- UNUSED, MUST BE 0 --
1 JSEBOTRY EQU   X'10'              READ = YES
1 JSEBOTWY EQU   X'08'              WRITE = YES
1 JSEBOTNU EQU   X'04'           -- UNUSED, MUST BE 0 --
1 JSEBOTNO EQU   X'02'              NO-ACCESS
1 JSEBOTUU EQU   X'01'           -- UNUSED, MUST BE 0 --
1 *
1 JSECRD   DC    CL10' '         CREATION DATE    - FROM DATE
1 JSECRT   DC    CL8' '                                 TIME
1 JSECRD2  DC    CL10' '                          - TO DATE
1 JSECRT2  DC    CL8' '                               TIME
1 *
1 JSEEXD   DC    CL10' '         EXPIRATION DATE  - FROM DATE
1 JSEEXT   DC    CL8' '                                 TIME
1 JSEEXD2  DC    CL10' '                          - TO DATE
1 JSEEXT2  DC    CL8' '                               TIME
1 *
1 JSEGUA   DC    B'00000000'     GUARDS SPECIFIED FLAG
1 JSEGUARS EQU   X'80'   7-7 S      READ-SPECIFIED
1 JSEGUAWS EQU   X'40'   6-6 S      WRITE-SPECIFIED
1 JSEGUAUN EQU   X'20'   5-5 R   -- UNUSED, MUST BE 0 --
1 JSEGUUNU EQU   X'10'   4-4 R   -- UNUSED, MUST BE 0 --
```

```
1 JSEGUANS EQU   X'08'   3-3 S      GUARDS=NONE SPECIFIED
1 JSEGUAYS EQU   X'04'   2-2 S      GUARDS=YES SPECIFIED
1 JSEGUAUU EQU   X'03'   1-0 R   -- UNUSED, MUST BE 0 --
1 JSEGUAR  DC    CL18' '            GUARDS-READ
1 JSEGUAW  DC    CL18' '            GUARDS-WRITE
1 *
1 JSEPWP   DC    B'00000000'     PW PROTECT FLAG -
1 JSEPWPRD EQU   X'80'   7-7 S      READ
1 JSEPWPWR EQU   X'40'   6-6 S      WRITE
1 JSEPWPUN EQU   X'20'   5-5 R   -- UNUSED, MUST BE 0 --
1 JSEPWPNO EQU   X'10'   4-4 S      NONE
1 JSEPWPNU EQU   X'0F'   3-0 R   -- UNUSED, MUST BE 0 --
1 *
1 JSEPROL  DC    B'00000000'     PROTECTION-LEVEL-FLAG
1 JSEPROL0 EQU   X'80'   7-7 S      LEVEL 0
1 JSEPROL1 EQU   X'40'   6-6 S      LEVEL 1
1 JSEPROL2 EQU   X'20'   5-5 S      LEVEL 2
1 JSEPROUU EQU   X'1F'   4-0 R   -- UNUSED, MUST BE 0 --
1 *
1 JSESELF1 DC    B'00000000'     SELECT FLAG 1 -
1 JSEACCRD EQU   X'80'   7-7 S      ACCESS = READ
1 JSEACCWR EQU   X'40'   6-6 S      ACCESS = WRITE
1 JSESHARY EQU   X'20'   5-5 S      SHARE  = YES
1 JSESHARN EQU   X'10'   4-4 S      SHARE  = NO
1 JSEMONY  EQU   X'08'   3-3 S      MONJV  = YES
1 JSEMONN  EQU   X'04'   2-2 S      MONJV  = NO
1 JSECJCY  EQU   X'02'   1-1 S      CJC    = YES
1 JSECJCN  EQU   X'01'   0-0 S      CJC    = NO
1 *
1 JSESELF2 DC    B'00000000'     SELECT FLAG 2 -
1 JSETLTI  EQU   X'80'   7-7 S      TIME-BASE = LOCAL
1 JSESF2UU EQU   X'7F'   6-0 R   -- UNUSED, MUST BE 0 --
1 *
1 JSEUNUS2 DC    X'000000'
1 *
1 JSESIZE  DC    A(0)            SIZE  - FROM
1 JSESIZE2 DC    A(0)                  - TO
1 *
1 JSEMGMCL DC    CL8' '          MANAGEMENT-CLASS
1 JSEFUT4  DC    60X'00'         SPACE FOR FUTURE USE, MUST BE 0
1 JSESPLLN EQU   *-JVSEL
```

LNKJV
Link job variables to JV-LINK entries

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **LNKJV** macro transfers information on the links between job variables and job variable link names (JV-LINK) from the JV-LINK table to a user area.

Macro call format and operand description

Operation	Operands
LNKJV	area [,length] $[,\left\{\begin{array}{l}\text{JVNAME=jvname}\\ \text{LINK=*jvlink}\end{array}\right\}]$ $,\text{ODSECT=}\left\{\begin{array}{l}\underline{\text{NO}}\\ \text{YES}\end{array}\right\}$ $,\text{VERSION=}\left\{\begin{array}{l}\underline{0}\\ 1\end{array}\right\}\ ,\text{MF=}\left\{\begin{array}{l}\underline{\text{S}}\\ \text{C}\\ \text{(E,..)}\\ \text{D}\\ \text{L}\end{array}\right\}\ ,\text{PREFIX=}\left\{\begin{array}{l}\underline{\text{IDJ}}\\ \text{pre}\end{array}\right\}$

area	Symbolic address of an area in the program to which the LNKJV information is to be transferred.

The area is supplied by LNKJV as follows:

ll_1	$jvlink_1$	$jvname_1$	ll_2	$jvlink_2$...
1 byte	8 bytes	up to 54 bytes		

```
<------------------------------------>
ll = total length of entry, up to 63 bytes
```

The end of the listed LNKJV information is identified by means of one byte; this last byte contains the value X'00' if all the LNKJV information has been transferred to the user area. If the user area was unable to accommodate all entries, the last byte contains the value X'01'.

length	Explicit length of the user area. If this operand is omitted, the implicit length of “area” is used. The user area must be at least 63 bytes long.
JVNAME=jvname	Fully or partially qualified path name of a permanent or temporary job variable whose link name is also to be placed in the defined area. The use of wildcards is allowed.
LINK=*jvlink	Valid link name of a job variable (as defined in DCLJV) whose complete path name is also to be placed in the defined area.
ODSECT	Specifies whether an additional DSECT for the output of a single item of LNKJV information is to be generated. This entry only makes sense with MF=D.
=<u>NO</u>	Default value; only one DSECT for the operand list of the LNKJV macro is generated.
=YES	Generates an additional DSECT for the output of a single item of LNKJV information (for format, see “area” or DSECT).
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.

VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7.
=1	The macro expansion is compatible with JV ≥ V10.0.

Notes concerning the DSECT

– Calling the LNKJV macro with the operand MF=D generates a DSECT.
  When VERSION=1 is specified, the DSECT is generated with a standard header.
– A DSECT is generated for the operand list of the LNKJV macro by default.
  If ODSECT=YES is specified, a DSECT is additionally generated for the output of an item of LNKJV information.

Return information and error flags

see page 229

DSECT

```
  LNKJV    LNKJV MF=D,VERSION=1,ODSECT=YES
1 ***********************************************************************
1 *        VERSION 320
1 ***********************************************************************
1 *       L N K J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                         C
1                INTNAME=LNKJV,INTCOMP=001
1 LNKJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=7,   VERSION=<PARAMETER VERSION>            *
1 ***********************************************************************
1          FHDR  MF=(C,IDJL)
2          DS    0A
2 IDJLFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJLIFID DS    0A              0   INTERFACE IDENTIFIER
2 IDJLFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJLFCT  DS    AL1             2   FUNCTION NUMBER
2 IDJLFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJLRET  DS    0A              4   GENERAL RETURN CODE
```

```
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJLSRET DS    0AL2            4   SUB RETURN CODE
2 IDJLSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJLR2OK EQU   X'00'               All correct, no additional info
2 IDJLR2NA EQU   X'01'               Successful, no action was necessary
2 IDJLR2WA EQU   X'02'               Warning, particular situation
2 IDJLSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJLRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 IDJLRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJLRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 IDJLRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 IDJLRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJLRAER EQU   X'04'               ALIGNMENT ERROR
2 IDJLRIER EQU   X'20'               INTERNAL ERROR
2 IDJLRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJLRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 IDJLRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJLRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 IDJLRWLR EQU   X'81'                    "     LONG       "
2 IDJLRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJLRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 IDJLRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 IDJLMRET DS    0AL2            6   MAIN RETURN CODE
2 IDJLMR2  DS    AL1             6   MAIN RETURN CODE 2
2 IDJLMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
```

```
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJLRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJLFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ***********************************************************************
1 *       END OF STANDARD HEADER - START SPECIAL LNKJV PARAMETERLIST    *
1 ***********************************************************************
1 IDJLHDRI       EQU   X'00290701',4
1 IDJLLINK      DS    CL8                   LINKNAME
1 IDJLNAME      DS    CL80                  JOBVARIABLE NAME
1 IDJLUNU2      DS    CL20                  UNUSED
1 IDJLADDR      DS    A                     AREA ADDRESS
1 IDJLSIZE      DS    H                     AREA SIZE
1 IDJLFLAG      DS    X                     INDICATOR
1 IDJLNOLI      EQU   X'80'                   LINK NOT SPECIFIED
1 IDJLNONA      EQU   X'10'                   JVNAME NOT SPECIFIED
1 IDJLECT       EQU   X'08'                   SET BY CMD PROCESSING
1 IDJLP2        EQU   X'04'                   P2 CALLER
1          DS    CL1                   ALIGNMENT
1 IDJLJVS       DS    A                     RESERVED
1 IDJLPLLN      EQU   *-LNKJV
1          SPACE
1 ***********************************************************************
1 *       L N K J V   O U T P U T                                      *
1 ***********************************************************************
1 IDJEL    DSECT
1 IDJELLN       DS    C                LAYOUT LENGTH
1 IDJELINK      DS    CL8              LINK NAME
1 IDJELJVN      DS    CL54             JV NAME (FULLY QUALIFIED)
1 IDJELLEN      EQU   *-IDJEL                LENGTH OF MACRO
1 ***********************************************************************
1          SPACE
```

ONEVT
Set condition for job variable event

General

Domain:	Job variables
Macro type:	Type S (with version=0): standard form/C/D/E/L form Type S (with version=1): standard form/C/D/E/L/M form, see section “The MF operand” on page 88

Macro description

The **ONEVT** macro is used to specify a condition and assign an event item. During program execution, a **POSSIG** is initiated by the system when the event “condition satisfied” or “catalog exported” occurs.
A maximum can be set for the number of **POSSIG** calls to be executed.
For more information on eventing, see the “Executive Macros” manual [4].

Macro call format and operand description

Operation	Operands
ONEVT	$\left\{\begin{matrix}\texttt{'cond.expr'}\\ \texttt{addr1}\\ \texttt{(r1)}\end{matrix}\right\}$,EIID=$\left\{\begin{matrix}\texttt{addr}\\ \texttt{(r)}\end{matrix}\right\}$ [,POST=$\left\{\begin{matrix}\texttt{val}\\ \texttt{(r)}\end{matrix}\right\}$] ,COUNT=$\left\{\begin{matrix}\underline{\texttt{1}}\\ \texttt{int}\\ \texttt{(r)}\end{matrix}\right\}$

Operation	Operands

ONEVT (cont.)	{ ,VERSION=0,MF={ S | (D,pre) | D | (E,...) | C | (C,pre) | L }
	 ,VERSION=1,MF={ S | C | D | E | L | M } [,PARAM={ addr | (r) }],PREFIX={ J | pre } ,MACID={ VSC | macid } }

'cond.expr' The condition is specified as a direct operand and must be enclosed in apostrophes. Because the apostrophe is used as a syntax character in assembler language, the following rules must be observed:
Each apostrophe within “cond.expression” must be doubled. The maximum length of “cond.expression” without the enclosing apostrophes is 127 bytes.
Special job variables are not allowed.

addr1 Symbolic address of a user program area containing the “cond.expression”.
The record format must be variable. The area starts with a 4-byte field, the first two bytes of which contain the length of the “cond.expression” in bytes + 4.

Example

```
SYMADR1      DC   Y(END-SYMADR1)
             DS   CL2
             DC   'conditional expression'
END          EQU  *
```

(r1)	Register containing the address of the “cond.expression”, which must be in the format described under “addr1” above. The “cond.expression” must not contain special job variables.
EIID	Names the event item ID made available to the user program by the TU eventing macro **ENAEI**. The **ENAEI** macro must be called before the **ONEVT** macro.
=addr	Symbolic address of the event item ID.
=(r)	Register containing the address of the event item ID.
POST	Specifies a 2-byte value which can be used at program level to identify **ONEVT**. It is passed to the user program in the two rightmost bytes of the post code. The post code is four bytes long and is passed to the program when events such as “condition satisfied” etc. occur.
=val	2-byte long Assembler constant of freely selectable format.
=(r)	Register containing the address of a 2-byte field in which the user stored the “value”.

Post code format:

Event-dependent indicator	Condition result	'ONEVT' identification “value”
1 byte	1 byte	2 bytes

Meanings of the first two bytes:

– event-dependent indicator:

X'14'	specifies that the event was caused by an ONEVT macro.

– condition result:
specifies the “reason” for the POSSIG specified by the system.

X'00':	condition satisfied
X'08':	catalog exported

If the POST operand is not present, no “value” for the ONEVT identification (i.e. X'00000000') is transferred.

COUNT	Specifies the maximum number of **POSSIG** macros that may be issued by the system (1 ≤ number ≤ 32767); default value = 1. The COUNT operand determines how often a program will be notified of an event.
=int	Number of **POSSIG** macros permitted for the “condition satisfied” event. When this number is reached, the **ONEVT** macro is deactivated. Irrespective of the COUNT operand entry, the **ONEVT** macro is deactivated immediately following a POSSIG macro caused by the event “catalog exported”, or by a DONEVT macro.
=(r)	Register containing the address of a half word in which the value of the COUNT operand is stored.
MF PREFIX MACID PARAM	For a description of the MF, PREFIX, MACID and PARAM operands, see page 88. The permitted values and the defaults for MF for this macro are indicated at the beginning of the macro description, those for PREFIX and MACID in the description of the VERSION operand.
VERSION	Specifies which version of BS2000 the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. With the C and D forms of this macro call a prefix (pre=1...3 letters) can be specified. Default: pre=ONE
=1	The macro expansion is compatible with JV ≥ V10.0. With the C form, D form or M form of the macro call a prefix (p=1 letter) can be specified, and with the C and D forms MACID also.

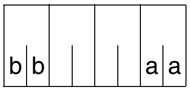
Function

Users can subject the result “condition satisfied” - about which they are informed by POSSIG - to BS2000-TU eventing as they see fit. For example, they may wait until “condition satisfied” has occurred (SOLSIG COND=) or a contingency process is executed asynchronously when “condition satisfied” applies (SOLSIG COID=...).

Return information and error flags

With VERSION=0:

R15



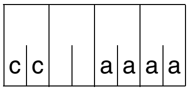
A return code relating to execution of the ONEVT macro (VERSION=0) is passed in register R15:
(aa = return switch, bb = secondary indicator)

X'bb'	X'aa'	Meaning
X'00'	X'00'	Execution terminated normally
X'00'	X'04'	Function not executed: invalid address for event item or condition or invalid COUNT value used
X'04'	X'04'	Function not executed: event item not found
X'08'	X'04'	Function not executed: invalid conditional expression
X'10'	X'04'	Function not executed: specified job variable cannot be accessed
	X'FF'	Function not executed: CJC not available in the system

With VERSION=1:

Following initialization of the standard header (when MF=S/L is specified in the call), the return information is made available at the symbolic address <PREFIX><MACID>RET (4 bytes).

Standard-header



A return relating to execution of the ONEVT macro (VERSION=1) is passed in the standard header:
(aaaa = main code, cc = subcode2)

X'cc'	X'aaaa'	Meaning
X'00'	X'0000'	Execution terminated normally
X'00'	X'0004'	Function not executed: invalid address for condition or event item, or illegal value for COUNT
X'04'	X'0004'	Function not executed: event item not found
X'08'	X'0004'	Function not executed: error in condition
X'10'	X'0004'	Function not executed: a specified job variable cannot be accessed
X'14'	X'0004'	Function not executed: insufficient memory
X'18'	X'0004'	Function not executed: eventing mechanism not available
	X'FFFF'	Function not executed: error during initialization of standard header. See page 229

In addition to the return codes named in VERSION=0 or 1, the general JV return codes for basic errors (e.g. invalid address for the operand list) may also appear. To find their meaning, see the relevant explanation under macro IDEJVS (page 229).

DSECT

```
  ONEVT    ONEVT MF=D,VERSION=1
1 ONEVT    MFCHK MF=D,                                                   C
1                SUPPORT=(C,D,E,L,M,S),                                  C
1                PREFIX=J,                                               C
1                MACID=VSC,                                              C
1                DMACID=VSC,                                             C
1                DNAME=ONEVTPL,                                          C
1                PARAM=,                                                 C
1                SVC=190
2 ONEVT    DSECT ,
2                *,##### PREFIX=J, MACID=VSC #####
1 JVSCFHDR   FHDR  MF=(C,JVSC),EQUATES=NO
2 JVSCFHDR DS    0A
2 JVSCFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 JVSCIFID DS    0A              0   INTERFACE IDENTIFIER
2 JVSCFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 JVSCFCT  DS    AL1             2   FUNCTION NUMBER
2 JVSCFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 JVSCRET  DS    0A              4   GENERAL RETURN CODE
2 JVSCSRET DS    0AL2            4   SUB RETURN CODE
2 JVSCSR2  DS    AL1             4   SUB RETURN CODE 2
2 JVSCSR1  DS    AL1             5   SUB RETURN CODE 1
2 JVSCMRET DS    0AL2            6   MAIN RETURN CODE
2 JVSCMR2  DS    AL1             6   MAIN RETURN CODE 2
2 JVSCMR1  DS    AL1             7   MAIN RETURN CODE 1
2 JVSCFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 JVSCUNU1   DS    CL3                         UNUSED
1 JVSCPCON   DS    X                           PARAM.-CONTROL
1 JVSCCORE   EQU   X'80'                       A(CONDITION) IN REGISTER
1 JVSCEREG   EQU   X'40'                       A(IDENTIFIER) IN REGISTER
1 JVSCPREG   EQU   X'20'                       POST IN REGISTER
1 JVSCCREG   EQU   X'10'                       COUNT IN REGISTER
1 JVSCPTPR   EQU   X'02'                       P2 CALLER
```

```
1 JVSCCOND   DS    A                               A(CONDITION)
1          ORG   JVSCCOND
1 JVSCCONR   DS    AL1                             REG. CONT. A(COND.)
1          DS    AL3                               NOT USED IN THIS CONTEXT
1 JVSCEIID    DS    A                              A(IDENTIFIER)
1          ORG   JVSCEIID
1 JVSCEIDR   DS    AL1                             REG. CONT. A(IDENTIFIER)
1          DS    AL3                               NOT USED IN THIS CONTEXT
1 JVSCPOST   DS    H                               POST-VALUE
1          ORG   JVSCPOST
1 JVSCPOSR   DS    AL1                             REG. CONT. POST-VALUE
1          DS    AL1                               NOT USED IN THIS CONTEXT
1 JVSCCNT    DS    H                               COUNT-VALUE
1          ORG   JVSCCNT
1 JVSCCNTR   DS    AL1                             REG. CONT. COUNT-VALUE
1          DS    AL1                               NOT USED IN THIS CONTEXT
1 JVSCJVS    DS    F                               RESERVED
1 JVSC#      EQU   *-JVSCFHDR                      LENGTH
1 *
1 * RETURNCODES
1 *
1 JVSCOK     EQU   X'00000000'                    SUCCESSFUL CALL
1 JVSCINAD   EQU   X'00010004'                    INVALID PARM.-LIST
1 JVSCINEI   EQU   X'04010004'                    INVALID IDENTIFIER
1 JVSCSYTE   EQU   X'08000004'                    SYNTAX ERROR IN CONDITION
1 JVSCJVNA   EQU   X'10000004'                    JV NOT ACCESSABLE
1 JVSCSYSE   EQU   X'14000004'                    SYSTEM ERROR ($GETMEM)
1 JVSCBOER   EQU   X'18000004'                    SYSTEM ERROR (BOURSES)
```

RELJV
Remove JV-LINK entry

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **RELJV** macro deletes one or all JV-LINK entries from the JV-LINK table. The entry to be deleted is selected via its link name.

Macro call format and operand description

Operation	Operands
RELJV	[LINK=*jvlink] ,MF= $\left\{\begin{array}{l}\underline{\text{S}}\\ \text{C}\\ \text{(E,..)}\\ \text{D}\\ \text{L}\end{array}\right\}$,PREFIX= $\left\{\begin{array}{l}\underline{\text{IDJ}}\\ \text{pre}\end{array}\right\}$

LINK=*jvlink	Valid link name of a job variable (as defined in DCLJV) which is to be deleted from the JV-LINK table. The link name may be up to 8 characters long, including the asterisk (*). If a link name consisting of blanks is specified, all entries are deleted from the JV-LINK table.
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.

Return information and error flags

see page 229

DSECT

```
  RELJV    RELJV MF=D
1 ***********************************************************************
1 *        VERSION 310
1 ***********************************************************************
1 *       R E L J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                        C
1                INTNAME=RELJV,INTCOMP=001
1 RELJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=33,  VERSION=1                              *
1 ***********************************************************************
1          FHDR   MF=(C,IDJR)
2          DS    0A
2 IDJRFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJRIFID DS    0A              0   INTERFACE IDENTIFIER
2 IDJRFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJRFCT  DS    AL1             2   FUNCTION NUMBER
2 IDJRFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJRRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJRSRET DS    0AL2            4   SUB RETURN CODE
2 IDJRSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJRR2OK EQU   X'00'               All correct, no additional info
2 IDJRR2NA EQU   X'01'               Successful, no action was necessary
2 IDJRR2WA EQU   X'02'               Warning, particular situation
2 IDJRSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
```

```
2 * CLASS A    X'00'            FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'    PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'            INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'    NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'    WAIT AND RETRY
2 *
2 IDJRRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJRRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJRRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJRRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJRRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJRRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJRRIER EQU   X'20'                INTERNAL ERROR
2 IDJRRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJRRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJRRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJRRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJRRWLR EQU   X'81'                     "     LONG       "
2 IDJRRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJRRTNA EQU   X'81'                SS TEMPORARILY NOT AVAILABLE
2 IDJRRDH  EQU   X'82'                SS IN DELETE / HOLD
2 *
2 IDJRMRET DS    0AL2             6   MAIN RETURN CODE
2 IDJRMR2  DS    AL1              6   MAIN RETURN CODE 2
2 IDJRMR1  DS    AL1              7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJRRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJRFHL  EQU   8                8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 **********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL RELJV PARAMETER LIST      *
1 **********************************************************************
1 IDJRHDRI      EQU   X'00292101',4
1 IDJRJVL       DS    CL8                     JV LINKNAME
1 IDJRFLG       DS    XL4                     FLAG RESERVED
1 IDJRJVS       DS    A                       RESERVED
1 IDJRPLLN      EQU   *-RELJV              LENGTH OF DSECT
1 **********************************************************************
1         SPACE
```

SETJV
Set job variable

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **SETJV** macro assigns a value to a job variable and stores it in an area of the user program.

Macro call format and operand description

Operation	Operands
SETJV	$\left\{\begin{array}{l}\texttt{jvid1}\\\texttt{(jvid1[,[start] [,länge]])}\end{array}\right\}$ $\left\{\begin{array}{l}\texttt{,area}\\\texttt{,EXPR=jvid2}\end{array}\right\}$ `[,PASS=kennwort]` $\texttt{,VERSION=}\left\{\begin{array}{c}\underline{\texttt{0}}\\\texttt{1}\end{array}\right\}\texttt{,PARMOD=}\left\{\begin{array}{c}\underline{\texttt{24}}\\\texttt{31}\end{array}\right\}\texttt{,MF=}\left\{\begin{array}{l}\underline{\texttt{S}}\\\texttt{C}\\\texttt{(E,..)}\\\texttt{D}\\\texttt{L}\end{array}\right\}\texttt{,PREFIX=}\left\{\begin{array}{c}\underline{\texttt{IDJ}}\\\texttt{pre}\end{array}\right\}$

jvid1 Identifies the job variable; jvid can be:

jvname	Fully qualified path name of a permanent or temporary job variable.
*jvlink	Valid job variable link name.

start Start position (first byte to be modified) in the JV value.

length	Number of characters to be set.
area	Address of an area in the user program containing the job variable value. The area must begin with a 4-byte length field whose first half word contains the length of the value + the 4-byte length field.
EXPR=jvid2	Specifies that the job variable “jvid” is to be set to the value of the job variable “jvid2”. The JV names permitted for “jvid2” are the same as those permitted for “jvid1” as well as those permitted for special job variables.
PASS=password	Read or write password.
MF PREFIX	For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format.
VERSION	Specifies which BS2000 version the macro expansion is to be compatible with.
=<u>0</u>	Default value; the macro expansion is compatible with JV ≤ V8.7. The operand MF=D/C, which generates a DSECT or CSECT respectively, is not supported in this version (see note on DSECT).
=1	The macro expansion is compatible with JV ≥ V10.0.
PARMOD	Controls macro expansion. Either the 24-bit or the 31-bit interface is generated. PARMOD is evaluated *only when VERSION=0 applies*. If PARMOD is not specified here, macro expansion is performed according either to the specification for the GPARMOD macro or to the default setting for the assembler (= 24-bit interface).
=<u>24</u>	The 24-bit interface is generated. Data lists and instructions use 24-bit addresses. (Address space ≤ 16 Mb.)
=31	The 31-bit interface is generated. Data lists and instructions use 31-bit addresses. (Address space ≤ 2 Gb.) Data lists start with the standard header.

Note concerning the operand list

Unlike at command level, a length of 0 may be specified in the operand list; this then represents the overall job variable length.

Notes concerning the DSECT

- Calling the SETJV macro with the operands MF=D and VERSION=1 generates a DSECT for the operand list of the SETJV macro (VERSION=1).
- A CSECT/DSEC for the macro with VERSION=0 is generated by calling the macro IDJSE [D][,prefix] [,PARMOD=24/31].

Return information and error flags

see page 229

DSECT

```
  SETJV    SETJV MF=D,VERSION=1
1 SETJV    $SETJV ,,EXPR=,PASS=NONE,ENCR=YES,MF=D,                         C
1                PARMOD=,VERSION=1,CALLER=USER,                            C
1                PREFIX=IDJ,JVTYPE=STRING,NULLSTR=NO
2 **********************************************************************
2 *        VERSION 203
2 ***********************************************************************
2 *       S E T J V   P A R A M E T E R   L I S T                      *
2 ***********************************************************************
2          #INTF REFTYPE=REQUEST,                                          C
2                INTNAME=SETJV,INTCOMP=002
2 SETJV    DSECT
2 ***********************************************************************
2 *       UNIT=41, FUNCTION=1,   VERSION=2  (V10.0)                    *
2 ***********************************************************************
2          FHDR   MF=(C,IDJS)
3          DS    0A
3 IDJSFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
3 *
3 IDJSIFID DS    0A              0   INTERFACE IDENTIFIER
3 IDJSFCTU DS    AL2             0   FUNCTION UNIT NUMBER
3 *                                  BIT 15    HEADER FLAG BIT,
3 *                                  MUST BE RESET UNTIL FURTHER NOTICE
3 *                                  BIT 14-12 UNUSED, MUST BE RESET
3 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
3 IDJSFCT  DS    AL1             2   FUNCTION NUMBER
3 IDJSFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
3 *
3 IDJSRET  DS    0A              4   GENERAL RETURN CODE
```

```
3 *
3 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
3 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
3 *
3 IDJSSRET DS    0AL2            4   SUB RETURN CODE
3 IDJSSR2  DS    AL1             4   SUB RETURN CODE 2
3 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
3 * Standard subcode2 values as defined by convention:
3 IDJSR2OK EQU   X'00'               All correct, no additional info
3 IDJSR2NA EQU   X'01'               Successful, no action was necessary
3 IDJSR2WA EQU   X'02'               Warning, particular situation
3 IDJSSR1  DS    AL1             5   SUB RETURN CODE 1
3 *
3 * GENERAL INDICATION OF ERROR CLASSES
3 *
3 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
3 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
3 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
3 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
3 * CLASS E    X'80' - X'82'   WAIT AND RETRY
3 *
3 IDJSRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
3 IDJSRPER EQU   X'01'               PARAMETER SYNTAX ERROR
3 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
3 IDJSRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
3 IDJSRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
3 IDJSRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
3 *
3 IDJSRAER EQU   X'04'               ALIGNMENT ERROR
3 IDJSRIER EQU   X'20'               INTERNAL ERROR
3 IDJSRCAR EQU   X'40'               CORRECT AND RETRY
3 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
3 IDJSRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
3 *                                  EXPLICITELY BY CREATE-SS
3 IDJSRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
3 *
3 IDJSRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
3 IDJSRWLR EQU   X'81'                    "     LONG       "
3 IDJSRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
3 *                                  BUT RETRY IS POSSIBLE
3 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
3 IDJSRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
3 IDJSRDH  EQU   X'82'               SS IN DELETE / HOLD
3 *
3 IDJSMRET DS    0AL2            6   MAIN RETURN CODE
3 IDJSMR2  DS    AL1             6   MAIN RETURN CODE 2
3 IDJSMR1  DS    AL1             7   MAIN RETURN CODE 1
3 *
```

```
3 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
3 *
3 IDJSRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
3 IDJSFHL  EQU   8                8   GENERAL OPERAND LIST HEADER LENGTH
3 *
2 **********************************************************************
2 *  END OF STANDARD HEADER. START OF SPECIAL SETJV PARAMETER LIST      *
2 **********************************************************************
2 IDJSHDRI      EQU   X'00290102',4
2 IDJSJV        DS    CL54                    JVNAME (JV TO BE SET)
2 IDJSPOS       DS    H                       SUBSTRING START POSITION
2 IDJSLEN       DS    H                       SUBSTRING LENGTH
2         DS    H                        RESERVED
2 IDJSPASS      DS    CL4                     PASSWORD
2         DS    XL12                     RESERVED
2 IDJSAR31      DS    A                       SET VALUE ADDR
2 IDJSLENV      DS    CL1                     LENGTH OF SET VALUE
2 *                                          ONLY USED BY CMD PROCESSING
2 IDJSJV2       DS    CL54                    JVNAME2 (JV TO BE READ)
2         DS    XL1                      ALIGNMENT
2 IDJSPOS2      DS    H                       SUBSTRING START POSITION
2 IDJSLEN2      DS    H                       SUBSTRING LENGTH
2 IDJSOP        DS    CL1                     RESERVED
2 IDJSF         DS    X                       FLAGS
2 IDJSFSE1      EQU   X'80'                   7-7 PRIV SETJV REQUEST
2 *                                          0=P1 CALLER,1=P2 CALLER
2 IDJSFRD       EQU   X'40'                   6-6 TYPE=READ
2 IDJSFWT       EQU   X'20'                   5-5 TYPE=WRITE
2 IDJSFUN       EQU   X'10'                   4-4 TYPE=UNLOCK
2 IDJSFSH       EQU   X'08'                   3-3 PROT=SHARE
2 IDJSFEX       EQU   X'04'                   2-2 PASSWD IS GIVEN
2 IDJSRW        EQU   X'02'                   1-1 RDPSTYPE=READ
2 IDJNOTF       EQU   X'01'                   0-0 NOTIF=NO
2 IDJSF1        DS    X                       FLAGS
2 IDJSENCR      EQU   X'80'                   7-7 0=YES, 1=NO
2 *                                          (ENCRYPTION)
2 IDJSF1NV      EQU   X'40'                   6-6 0=NO,  1=YES
2 *                                          (NUMERIC VALUE)
2 IDJSJVIX      EQU   X'20'                   5-5 0=NO,  1=YES
2 *                                          (JVID (1 OR 2) INDEXED)
2 *                                      4-4 RESERVED FOR MONJV-HANDLER
2 IDJSECT       EQU   X'08'              3-3 1=SET BY CMD PROCESSING
2 IDJSF1BV      EQU   X'04'                   2-2 0=NO,  1=YES
2 *                                          (BOOLEAN VALUE)
2 IDJSNSTR      EQU   X'02'                   1-1 0=NO,  1=YES
2 *                                          (NULLSTRING DEFINED)
2         DS    XL1                      ALIGNMENT
2 IDJSCTAD      DS    A                       CATALOG ENTRY ADDR
```

```
2 IDJSJVER      DS    CL54                    JV IN ERROR
2         DS    XL2                      RESERVED
2 IDJSJVS       DS    A                       RESERVED
2 IDJSPLLN      EQU   *-SETJV              LENGTH OF DSECT
2 **********************************************************************
2         SPACE
```

STAJV
Output job variable attributes

General

Domain:	Job variables
Macro type:	Type S (standard form/C/D/E/L form) see section “The MF operand” on page 88

In the C and D forms of the macro, a prefix (PREFIX = pre, where pre is 1..3 letters) can be specified (see section “The PREFIX operand” on page 89).

Default value:	PREFIX=IDJ

Macro description

The **STAJV** macro transfers the attributes of a job variable to a user area or creates a list of job variable names in this area.

Macro call format and operand description

Operation	Operands
STAJV	[jvid]
	,area [,size]
	,OUTPUT=$\left\{\begin{array}{c}\underline{\texttt{OLD}}\\ \texttt{NEW}\end{array}\right\}$,LIST=$\left\{\begin{array}{c}\underline{\texttt{OLD}}\\ \texttt{NEW}\end{array}\right\}$,TIMBASE=$\left\{\begin{array}{c}\underline{\texttt{*UTC}}\\ \texttt{*LTI}\end{array}\right\}$
	,ODSECT=$\left\{\begin{array}{c}\underline{\texttt{NO}}\\ \texttt{YES}\end{array}\right\}$[,SELADDR=auswahl],SORT = $\left\{\begin{array}{l}\underline{\texttt{*JVNAM}}\\ \texttt{NO}\end{array}\right\}$
	,VERSION=$\left\{\begin{array}{c}\underline{\texttt{0}}\\ \texttt{1}\\ \texttt{2}\\ \texttt{3}\\ \texttt{4}\end{array}\right\}$,PARMOD=$\left\{\begin{array}{c}\underline{\texttt{24}}\\ \texttt{31}\end{array}\right\}$,MF=$\left\{\begin{array}{l}\underline{\texttt{S}}\\ \texttt{C}\\ \texttt{(E,..)}\\ \texttt{D}\\ \texttt{L}\end{array}\right\}$
	,PREFIX=$\left\{\begin{array}{c}\underline{\texttt{IDJ}}\\ \texttt{pre}\end{array}\right\}$

jvid	Identifies the job variable whose attributes are to be transferred to the user area. This is not possible for special job variables. jvid can be:	
	jvname	A fully or partially qualified path name of a permanent or temporary job variable. The use of wildcards is allowed for versions > 0. The attributes of a job variable are transferred to the user area only when a fully qualified name is specified. Otherwise, a list of job variable names is placed in the user area. If a user specifies a different user ID, only the names of those job variables which the catalog entry identifies as allowing access by other IDs are transferred to the user area. This is SHARE=YES, and R or W for GROUP or OTHERS if basic ACL protection is in effect.
	*jvlink	A valid job variable link name.
	temp	A name list is output containing all of the temporary job variables created for the job. “temp” stands for the special character defined with the system parameter TEMPFILE, used to identify temporary files and job variables (if necessary, ask systems support which character has been defined).

area — Specifies the user area to which the information is to be transferred (see “Function” below).

size — Specifies the length of the area in bytes.
Up to VERSION=3 the maximum length is 32767 bytes.
The length of the area is dependent on the output format:

OUTPUT=OLD :
– size ≥ 60 bytes, where 60 bytes are set as default value;

OUTPUT=NEW:
– size ≥ 80 bytes, where 80 bytes are set as default value (when VERSION=2 is specified (JV V10.0))
– size ≥ 120 bytes, where 120 bytes are set as default value (when VERSION=3 is specified (JV V11.0 und V11.2))
– size ≥ 160 bytes, where 160 bytes are set as default value (when VERSION=4 is specified (JV ≥ V12.0))

OUTPUT	Specifies the output format for a job variable whose name was specified in *fully qualified* form. *This specification is only permitted for Version 2 and higher and when MF=L or MF=S is specified.*
=OLD	Default value; the requested information contains only the fixed part of the catalog entry, but *without* the name of the job variable (60 bytes). A DSECT for this output format is generated with the IDJE macro (see also note on DSECT).
=NEW	The information is requested in the new format. – If VERSION=2 is specified, the information is output in JV V10.0 layout (80 bytes). – If VERSION=3 is specified, the information is output in JV V11.0 or JV V11.2 layout (120 bytes). – If VERSION=4 is specified, the information is output in JV ≥ V12.0 layout (160 bytes).
LIST	Determines the output format in the user area. This operand only takes effect when Version ≥ 1 and the path name is specified in *partially qualified* form (see “Function” below).
=OLD	Default value; the old output format (without :catid:$userid.) is supported.
=NEW	Outputs the complete path name (with :catid:$userid.).
TIMBASE	Determines the time base to be used for the creation date and the deletion date. This operand only takes effect when VERSION=4 and OUTPUT=NEW are specified.
=*UTC	Default value; UTC time (global time) is to be used as the time base.
=*LTI	Specifies that the local time set in the calling system is to be used as the time base.
ODSECT	Specifies for which output format a DSECT is to be generated. *Permitted only when MF=D is specified.*
=NO	Default value; the DSECT for the old output format can be generated with the IDJE macro (see OUTPUT=OLD).
=YES	The DSECT is generated for the new output format (80/120/160 bytes; see OUTPUT=NEW).

SELADDR

Limits the jobset specified by the fully or partially qualified path name jvid, with further selection criteria. If the operand is not specified, the IDJFJVS field is set to zero, and no further selection takes place.
Values can only be specified in versions ≥ 2.

=selection

Symbolic address of the parameter list which was created with the desired selection criteria by calling the JVSEL macros (see page 136ff). See also “Function” on page 176.

SORT

Determines the sorting of catalog entries/ path names in the output.

=<u>*JVNAM</u>

The catalog entries/ path names are sorted alphabetically for output; default.

=NO

The catalog entries/ path names are output in the order that they appear in the catalog.

MF
PREFIX

For a description of the MF and PREFIX operands, see page 88. Their permitted values are indicated at the beginning of the macro description and in the macro call format. The MF=D/C operand used to generate a DSECT or CSECT is only supported in VERSION=2 and higher (see the DSECT note).

VERSION

Specifies which BS2000 version the macro expansion is to be compatible with.

=<u>0</u>

Is the default: the macro expansion is compatible with JV < V8.7. The following specifications are not supported yet in this version:
- MF=C/D (see the DSECT note)
- LIST=NEW
- OUTPUT=NEW
- SELADDR=

=1

The macro expansion is compatible with JV V8.7. The following specifications are not supported yet in this version:
- MF=C/D (see the DSECT note)
- OUTPUT=NEW
- SELADDR=

=2

The macro expansion is compatible with JV V10.0.

=3

The macro expansion is compatible with JV V11.0 and V11.2.

=4

The macro expansion is compatible with JV ≥ V12.0.

PARMOD	Controls macro expansion. Either the 24-bit interface or the 31-bit interface is generated. *PARMOD is evaluated only with VERSION=0.* If PARMOD is not specified, macro expansion is performed according to the specification for the GPARMOD macro or according to the default setting for the assembler (= 24-bit interface).
=<u>24</u>	The 24-bit interface is generated. Data lists and instructions use 24-bit addresses. (Address space ≤ 16 Mb.)
=31	The 31-bit interface is generated. Data lists and instructions use 31-bit addresses. (Address space ≤ 2 Gb.) Data lists start with the standard header.

Function

If a *fully qualified* job variable name is specified, the checking information is transferred from the JV catalog entry to the specified area. The OUTPUT operand determines the format of the output here.
The DSECT for the catalog entry can be generated with MF=D. The password field is set to binary zero. If the specified area is too small to accommodate the information, the program receives the return code X'0490' in the rightmost two bytes of register 15.
If the specified job variable name is *not fully qualified* (partially qualified or containing wildcards), or if the special character for temporary job variables is specified, a list of the job variable names is transferred to the specified area. The format of the information transferred depends on what is specified in the LIST operand:

```
                   LIST=OLD            LIST=NEW

name length1       1 byte              1 byte

jvname1,           1–41 characters     14–54 characters    jvname,
without catid,                                             with catid, userid
userid

......

name lengthn       1 byte              1 byte
jvnamen            1–41 characters     14–54 characters
without                                                    with catid, userid
catid, userid
End                1 byte(X'00'/X'01') 1 byte(X'00'X'01')
```

“namelength” specifies the length of the associated job variable name (plus 1-character length field). The proper end of the list is marked by X'00'. If the area is not long enough to accommodate all the job variable names, the last byte is set to X'01'.
When a partially qualified job variable name is specified, an additional corresponding value (IDJPQFN=X’20’) is returned in the parameter list of the STAJV call.

Additional selection of job variables

The set of job variable names to be transferred can be limited to the job variables with specific characteristics. When a fully qualified job variable name is entered, the transfer of catalog information can be made dependent on specific characteristics.
The desired selection criteria are specified in a JVSEL macro call. In the STAJV call, the symbolic address of the created JVSEL parameter list must then be specified in the operand SELADDR. If SELADDR is not specified, the field IDJFJVS is set to zero (X'00000000'= no selection according to selection criteria).

Note

If no operands are specified, a list is output containing all the job variables included in the standard catalog of the user ID under which the job executes.

Notes concerning the DSECT

– Calling the STAJV macro with the operands MF=D and VERSION=2/3/4 generates a DSECT for the operand list of the STAJV macro (VERSION=2/3/4). If ODSECT=YES is specified, an additional DSECT is generated for output of a catalog entry.
– The following applies to VERSION=0 or VERSION=1:
  The macro call IDJE [D][,prefix] generates a CSECT/DSECT for the catalog entry.
  The macro call IDJST [D][,prefix][,PARMOD=24/31][,VERSION=0/1] generates a CSECT/DSECT for the operand list of the STAJV macro.

Return information and error flags

see page 229

DSECT

```
  STAJV    STAJV MF=D,VERSION=4,ODSECT=YES
1 ***********************************************************************
1 *        VERSION 400
1 ***********************************************************************
1 *       S T A J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                         C
1                INTNAME=STAJV,INTCOMP=005
1 STAJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=2,   VERSION=<PARAMETER VERSION>            *
1 ***********************************************************************
1          FHDR  MF=(C,IDJF)
2          DS    0A
2 IDJFFHE  DS    0XL8             0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJFIFID DS    0A               0   INTERFACE IDENTIFIER
2 IDJFFCTU DS    AL2              0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJFFCT  DS    AL1              2   FUNCTION NUMBER
2 IDJFFCTV DS    AL1              3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJFRET  DS    0A               4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJFSRET DS    0AL2             4   SUB RETURN CODE
2 IDJFSR2  DS    AL1              4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJFR2OK EQU   X'00'                All correct, no additional info
2 IDJFR2NA EQU   X'01'                Successful, no action was necessary
2 IDJFR2WA EQU   X'02'                Warning, particular situation
2 IDJFSR1  DS    AL1              5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'            FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'    PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'            INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'    NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'    WAIT AND RETRY
```

```
2 *
2 IDJFRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 IDJFRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJFRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 IDJFRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 IDJFRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJFRAER EQU   X'04'               ALIGNMENT ERROR
2 IDJFRIER EQU   X'20'               INTERNAL ERROR
2 IDJFRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJFRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 IDJFRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJFRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 IDJFRWLR EQU   X'81'                    "     LONG       "
2 IDJFRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJFRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 IDJFRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 IDJFMRET DS    0AL2            6   MAIN RETURN CODE
2 IDJFMR2  DS    AL1             6   MAIN RETURN CODE 2
2 IDJFMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJFRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJFFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 ***********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL STAJV PARAMETER LIST      *
1 ***********************************************************************
1 IDJFHDRI      EQU   X'00290205',4
1 IDJAR31       DS    F                        AREA ADDRESS(31 BIT FORMAT)
1 IDJSIZE       DS    F                        AREA SIZE
1 IDJFLAGS      DS    X                        FLAGS
1 IDJPASSW      EQU   X'00'                    7-7 S NOT USED (DEL. V9.5)
1 IDJNAREA      EQU   X'40'                    6-6 S ADDR NOT GIVEN
1 *                                             SET BY JVSTAEX
1 IDJPQFN       EQU   X'20'                    5-5 S PART. QUAL. JVNAME
1 *                                             SET BY JVSTAEX
1 *                                             FOR CMD PROCESSING
1 IDJFECT       EQU   X'10'                    4-4 S SET BY CMD PROCESSING
1 IDJLNEW       EQU   X'08'                    3-3 S OUTPUT JVNAME LONG
```

```
1 IDJFCE        EQU   X'04'                    2-2 S NEW CE OUTPUT V10
1 IDJFP2        EQU   X'02'                    1-1 S P2 CALLER
1 IDJFCE11      EQU   X'01'                    0-0 S NEW CE OUTPUT V11
1 IDJFLAG1      DS    X                        FLAGS
1 IDJFCE12      EQU   X'80'                    7-7 S NEW CE OUTPUT V12
1 IDJFTLTI      EQU   X'40'                    6-6 S TIME-BASE = LOCAL
1 IDJFSORT      EQU   X'20'                    5-5 S SORT      = NO
1          DS    X                        RESERVED
1          DS    CL16                     RESERVED
1 IDJFJV        DS    CL80                     JV-NAME
1          DS    CL41                     RESERVED
1 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1 * THE FOLLOWING FIELD IS USED ONLY FOR CMD PROCESSING *
1 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
1 IDJSRTAD      DS    A                        ADDRESS OF JVNAME
1 IDJFJVS       DS    A                        ADDRESS OF SELECTION PL
1 IDJFPLLN      EQU   *-STAJV               LENGTH OF DSECT
1          SPACE
1 ***********************************************************************
1 *       S T A J V   O U T P U T                                       *
1 ***********************************************************************
1 IDJES    DSECT
1 IDJEDMS       DS    CL1         0 DMS INDICATOR
1 IDJESJV       EQU   X'04'         S SHARE = YES
1 IDJEROA       EQU   X'08'         S READ ONLY ACCESS
1 IDJERPP       EQU   X'10'         S READ PASSWORD SPECIFIED
1 IDJEWPP       EQU   X'20'         S WRITE PASSWORD SPECIFIED
1 IDJEDM2       DS    CL1         1 DMS INDICATOR2
1 IDJENSH       EQU   X'80'         S NON-SHARE FLAG
1 IDJECON       EQU   X'40'         S CONVERTED CE
1 IDJENCR       EQU   X'01'         S PASSWORD IS ENCRYPTED
1 IDJERD        DS    CL4         2 READ PASSWORD
1 IDJEWRT       DS    CL4         6 WRITE PASSWORD
1 IDJECRD       DS    0CL14      10 CREATION DATE
1 IDJECRY       DS    CL4           YEAR
1 IDJECRM       DS    CL2           MONTH
1 IDJECRT       DS    CL2           DAY
1 IDJECRH       DS    CL2           HOURS
1 IDJECRI       DS    CL2           MINUTES
1 IDJECRS       DS    CL2           SECONDS
1 IDJEEXD       DS    0CL14      24 EXPIRATION DATE
1 IDJEEXY       DS    CL4           YEAR
1 IDJEEXM       DS    CL2           MONTH
1 IDJEEXT       DS    CL2           DAY
1 IDJEEXH       DS    CL2           HOURS
1 IDJEEXI       DS    CL2           MINUTES
1 IDJEEXS       DS    CL2           SECONDS
1 IDJEACL       DS    CL1        38 ACCESS CONTROL LIST
```

```
1 IDJEMAC       EQU   X'80'         S BASIC ACL PRESENT
1 IDJEGACT      EQU   X'40'         S GUARD PRESENT
1 IDJEURD       EQU   X'20'         S USER: READ-PRIVILEG
1 IDJEUWR       EQU   X'10'         S USER: WRITE-PRIVILEG
1 IDJEGRD       EQU   X'08'         S GROUP: READ-PRIVILEG
1 IDJEGWR       EQU   X'04'         S GROUP: WRITE-PRIVILEG
1 IDJEORD       EQU   X'02'         S OTHERS: READ-PRIVILEG
1 IDJEOWR       EQU   X'01'         S OTHERS: WRITE-PRIVILEG
1 IDJEDIV       DS    CL1        39 ADDITIONAL CHARASTERISTICS
1 IDJETYP       EQU   X'80'         S NUMERIC-TYPE
1 IDJEMJV       EQU   X'40'         S MONJV
1 IDJETEMP      EQU   X'20'         S TEMPORARY INDICATOR
1 IDJENSTD      EQU   X'10'         S NULL STRING DEFINED
1 IDJEBLID      EQU   X'08'         S BOOLEAN-TYPE
1 IDJECAT       DS    CL4        40 GUARD: CATID
1 IDJEREAD      DS    CL18       44 GUARD: READ
1 IDJEWRIT      DS    CL18       62 GUARD: WRITE
1 IDJEMANC      DS    CL8        80 MANAGEMENT CLASS
1 IDJEVLL       DS    CL2        88 JV VALUE LENGTH
1 IDJECLN       DS    CL1        90 CATID LENGTH
1 IDJEULN       DS    CL1        91 USERID LENGTH
1 IDJEJLN       DS    CL1        92 JV NAME LENGTH
1 IDJEPLN       DS    CL1        93 PATHNAME LENGTH
1 IDJENAM       DS    CL54       94 JV NAME
1         DS    CL12     148 RESERVED
1 IDJELEN       EQU   *-IDJEDMS      LENGTH OF FIXED PORTION
1 ***********************************************************************
1         SPACE
```

TERM
Terminate program and procedure step

General

Domain: Job variables

Macro type: Type S (standard form/E form/L form)

Macro description

The TERM macro performs the following functions:

- terminate program (default value)
- terminate program and procedure step (operand UNIT=STEP)
- take memory dump (DUMP operand)
- transfer return code to program-monitoring job variable (URETCD operand).

All input/output operations initiated by the program before execution of the macro are completed prior to program termination.

Macro call format and operand description

Operation	Operands
TERM	$\texttt{UNIT=}\left\{\begin{array}{l}\underline{\texttt{PRGR}}\\ \texttt{STEP}\end{array}\right\}\texttt{,DUMP=}\left\{\begin{array}{l}\underline{\texttt{N}}\\ \texttt{Y}\end{array}\right\}\texttt{,MODE=}\left\{\begin{array}{l}\underline{\texttt{N[ORMAL]}}\\ \texttt{A[BNORMAL]}\end{array}\right\}$ $\texttt{[,URETCD=}\left\{\begin{array}{l}\texttt{addr}\\ \texttt{(r)}\\ \texttt{code}\end{array}\right\}\texttt{]}$ $\texttt{,MF=}\left\{\begin{array}{l}\texttt{S}\\ \texttt{(E,...)}\\ \texttt{L}\end{array}\right\}$

UNIT Determines whether a distinction according to operating modes is to be made when the program terminates.

=<u>PRGR</u> Default value; the program is terminated.

=STEP	Terminates the program, with further action being determined by the operating mode in which the program executed. – Interactive mode: If the program was called in a non-S procedure, the system additionally branches to the next SET-JOB-STEP, EXIT-JOB or CANCEL-PROCEDURE command. If the program was called in an S procedure, the system additionally initiates SDF-P error recovery. – Batch mode (ENTER procedure): The system additionally branches to the next SET-JOB-STEP, EXIT-JOB, ABEND or LOGOFF command. *Note* The following settings are recommended: UNIT=PRGR with MODE=NORMAL UNIT=STEP with MODE=ABNORMAL
DUMP	Determines whether a memory dump is to be taken.
=<u>N</u>	Default value; no memory dump is taken.
=Y	A memory dump is taken provided no MODIFY-TEST-OPTIONS command has been specified with DUMP=NO.
MODE	Determines the termination mode of the program.
=<u>N[ORMAL]</u>	Default value; the program is to terminate normally. If a program-monitoring job variable is defined, its status indicator is set to C'$T␣'.
=A[BNORMAL]	The program is to terminate abnormally. The message "`.... ABNORMAL PROGRAM TERMINATION (&00)`" is output. `(&00)= NRT0001` if UNIT=PRGR was specified `(&00)= NRT0101` if UNIT=STEP was specified For users of job variables: If a program-monitoring job variable has been defined, its status indicator is set to $A. *Note* see the UNIT operand.

URETCD	This value is passed as the return code to the program-monitoring job variable (left-justified, bytes 4-7). If this operand is omitted, the value C'␣␣␣␣' is passed to the program-monitoring job variable. The operand is ignored if no program-monitoring job variable has been defined.
=code	Specifies an alphanumeric value, 1 to 4 bytes long, in decimal (C'cccc') or hexadecimal (X'xxxxxxxx') form.
=addr	Relative virtual address of a 4-character alphanumeric value.
=(reg)	Register containing a 4-character alphanumeric value. If URETCD=(reg) is specified, register R0 is destroyed.
MF	For a general description of the MF operand, its associated operand values and any succeeding operands, see the section “The MF operand” on page 88. The permitted MF values are indicated at the beginning of the macro description under the macro type, and in the macro call format.

Function

When this macro is executed the following occurs:

– All files assigned to the program are closed.
– The memory assigned to the program is released.
– The BBS operations list and the entries in the BBS program table are released.
– If an STXIT routine has been defined for the TERM event class, it is activated.
– AIDSYS is called with the event “TERM”.
– In the device table, bytes 8-30 are deleted for each device released. The first byte in the operation list is set to X'FF'. In the program table entry, the program start address is set to 0 (4 bytes).
– The system then switches to command mode.

Notes

– The operand is ignored if an invalid address is specified in the URETCD operand.

– Register R1 contains the address of the data area.
If the data area address is invalid or if incorrect operands were specified, TERM UNIT=STEP,MODE=ABNORMAL,DUMP=Y is executed and the following error message is issued:

```
%.... ABNORMAL PROGRAM TERMINATION NRT0601
```

– Calling the TERM macro with the operand DUMP=Y produces the message
`'PROCESSING INTERRUPTED AT...)'`

  Whether or not a dump is taken depends on the value of the DUMP operand in the MODIFY-TEST-OPTIONS command. When DUMP=STD (default value) is set, TERM causes one of the following messages to be issued:

  – In interactive mode:

    ```
    'DUMP DESIRED ? REPLY (Y=YES, N=NO)'
    ```

    Whether or not a dump is output depends on the user's response.

  – In batch mode and in procedures:

    ```
    'SYSTEM REGULATIONS PROHIBIT DUMP'
    ```

    No dump is generated.

TIMJV
Modify a job monitoring JV

General

Domain:	Job variables
Macro type:	Type S (standard form/C form/D form/E form /L form)

A PREFIX (pre = 1..3 characters) can be specified for the C form and D form of the macro call (see section “The PREFIX operand” on page 89);

Default setting:	PREFIX=IDJ

Macro description

The following elements in the system section of a job monitoring job variable can be modified at the program level with the TIMJV macro (see also the command MODIFY-MONJV):

– Time stamp (operand TIMESTAMP)

– Name of the job to be monitored (operand DESCRIPTOR)

– Job specific information (operand USER-INFORMATION)

The format and position of the elements in the system section are described in the section “Values for monitoring job variables” on page 54.

Macro call format and operand description

Operation	Operands
TIMJV	[MONJV=jvid]
	,TIMESTAMP={ *UNCHANGED / *SET }
	,DESCRIPTOR={ *UNCHANGED / jobname }
	,INFO={ *UNCHANGED / jobinfo }
	[,PASS=password]
	,MF={ S / C / (E,..) / D / L } ,PREFIX={ IDJ / pre } ,VERSION=1

MONJV=jvid — Name of the monitoring job variable. When no job variable is specified, the macro affects the monitoring job variable of the job in which it was called.

TIMESTAMP	Specifies if a time stamp is to be set for job monitoring.
=*UNCHANGED	Default: No time stamp is recorded.
=*SET	A time stamp is recorded in GMT time. Format: yyyy-mm-ddhhmmss.
DESCRIPTOR	Specifies if the name of the job to be monitored is set..
=*UNCHANGED	Default: No value is recorded.
=jobname	Name of the job to be monitored (max. 8 byte long string).

INFO	Specifies if job-specific information is to be set.
=<u>*UNCHANGED</u>	Default: No value is recorded.
=jobinfo	Job-specific information (max. 58 byte long string).
PASS=password	Read or write password of the job variable.
MF PREFIX	See page 88 for descriptions or the MF and PREFIX operands. The valid values are shown at the beginning of the macro description and can be obtained from the call format.
VERSION = <u>1</u>	Default. The macro expansion is compatible with JV ≥ V13.0B.

Note

The operands TIMESTAMP, DESCRIPTOR and INFO may only be specified when generating macros with MF=S or MF=L

Return information and error flags

See page 229.

DSECT

```
  TIMJV    TIMJV MF=D
1 ***********************************************************************
1 *        VERSION 312
1 ***********************************************************************
1 *       T I M J V   P A R A M E T E R   L I S T                       *
1 ***********************************************************************
1          #INTF REFTYPE=REQUEST,                                         C
1                INTNAME=TIMJV,INTCOMP=001
1 TIMJV    DSECT
1 ***********************************************************************
1 *       UNIT=41, FUNCTION=34,  VERSION=1                              *
1 ***********************************************************************
1          FHDR   MF=(C,IDJT)
2          DS    0A
2 IDJTFHE  DS    0XL8             0   GENERAL PARAMETER AREA HEADER
2 *
2 IDJTIFID DS    0A               0   INTERFACE IDENTIFIER
2 IDJTFCTU DS    AL2              0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
```

```
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 IDJTFCT  DS    AL1              2   FUNCTION NUMBER
2 IDJTFCTV DS    AL1              3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 IDJTRET  DS    0A               4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 IDJTSRET DS    0AL2             4   SUB RETURN CODE
2 IDJTSR2  DS    AL1              4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 IDJTR2OK EQU   X'00'                All correct, no additional info
2 IDJTR2NA EQU   X'01'                Successful, no action was necessary
2 IDJTR2WA EQU   X'02'                Warning, particular situation
2 IDJTSR1  DS    AL1              5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 IDJTRFSP EQU   X'00'                FUNCTION SUCCESSFULLY PROCESSED
2 IDJTRPER EQU   X'01'                PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 IDJTRFNS EQU   X'01'                CALLED FUNCTION NOT SUPPORTED
2 IDJTRFNA EQU   X'02'                CALLED FUNCTION NOT AVAILABLE
2 IDJTRVNA EQU   X'03'                INTERFACE VERSION NOT SUPPORTED
2 *
2 IDJTRAER EQU   X'04'                ALIGNMENT ERROR
2 IDJTRIER EQU   X'20'                INTERNAL ERROR
2 IDJTRCAR EQU   X'40'                CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 IDJTRECR EQU   X'41'                SUBSYSTEM (SS) MUST BE CREATED
2 *                                   EXPLICITELY BY CREATE-SS
2 IDJTRECN EQU   X'42'                SS MUST BE EXPLICITELY CONNECTED
2 *
2 IDJTRWAR EQU   X'80'                WAIT FOR A SHORT TIME AND RETRY
2 IDJTRWLR EQU   X'81'                     "     LONG       "
2 IDJTRWUR EQU   X'82'                WAIT TIME IS UNCALCULABLY LONG
2 *                                   BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 IDJTRTNA EQU   X'81'                SS TEMPORARILY NOT AVAILABLE
```

```
2 IDJTRDH  EQU   X'82'                SS IN DELETE / HOLD
2 *
2 IDJTMRET DS    0AL2            6   MAIN RETURN CODE
2 IDJTMR2  DS    AL1             6   MAIN RETURN CODE 2
2 IDJTMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 IDJTRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 IDJTFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 **********************************************************************
1 *  END OF STANDARD HEADER. START OF SPECIAL TIMJV PARAMETER LIST      *
1 **********************************************************************
1 IDJTHDRI      EQU   X'00292201',4
1 IDJTJV        DS    CL54                     MONJV
1 IDJTFLG       DS    XL1                      FLAG
1 IDJTTIME      EQU   X'80'                    7-7 1=SET TIMESTAMP 0=UNCH
1 IDJTDEUN      EQU   X'40'                    6-6 1=DESCRIPTOR UNCHANGED
1 *                                                0=DESCRIPTOR GIVEN
1 IDJTINUN      EQU   X'20'                    5-5 1=INFO UNCHANGED
1 *                                                0=INFO GIVEN
1 IDJTRES1      DS    XL1                      RESERVED
1 IDJTPASS      DS    CL4                      PASSWORD
1          DS    XL12
1 IDJTDESC      DS    CL8                      DESCRIPTOR
1 IDJTINFO      DS    CL58                     INFO
1 IDJTRES2      DS    XL2                      RESERVED
1 IDJTJVS       DS    A                        RESERVED
1 IDJTPLLN      EQU   *-TIMJV               LENGTH OF DSECT
1 **********************************************************************
1          SPACE
```

5 Examples

Example 1 (management of job variables)

This example illustrates the use of the commands for the management of job variables:

```
/show-jv-attr ──────────────────────────────────────────────────── (1)
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS1
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS2
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS3
%SUM    00004 JV'S; JV-VALUE = 00000000 BYTES
/create-jv jv=jv.perm.error2 ───────────────────────────────────── (2)
/show-jv-attributes jv.perm.error* ─────────────────────────────── (3)
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR2
%SUM    00002 JV'S; JV-VALUE = 00000000 BYTES
/modify-jv jv=jv.perm.error2,set-val=c'No Error' ───────────────── (4)
/show-jv-attr jv=jv.perm.error2,inf=*all-attr ──────────────────── (5)
%0000008 :4V05:$COGNITAS.JV.PERM.ERROR2
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:14:52  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000008 BYTES
/show-jv jv=jv.perm.error2 ─────────────────────────────────────── (6)
%No Error
```

(1) All permanent job variables are output.

(2) The job variable JV.PERM.ERROR2 is created with default values.

(3) All job variables beginning with the character string “JV.PERM.ERROR” are output.

(4) The job variable JV.PERM.ERROR2 is given the contents 'No Error'.

(5) All attributes of the job variable JV.PERM.ERROR2 are output.

(6) The contents of the job variable JV.PERM.ERROR2 are output.

```
/mod-jv-attr jv=jv.perm.error2,prot=(write-pass=c'c5aq') ─────────── (7)
/modify-jv jv=jv.perm.error2,set-val=c'write error' ──────────────── (8)
%  JVS04B1 PASSWORD NOT SPECIFIED. COMMAND REJECTED
/add-pass password=c'c5aq' ─────────────────────────────────────── (9)
/modify-jv jv=jv.perm.error2,set-val=c'write error'
/show-jv jv=jv.perm.error2 ─────────────────────────────────────── (10)
%write error
/show-jv-attr jv=jv.perm.error2,inf=*all-attr ──────────────────── (11)
%0000011 :4V05:$COGNITAS.JV.PERM.ERROR2
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:17:41  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = YES
SUM    00001 JV'S; JV-VALUE = 00000011 BYTES
/create-jv jv=#jv.temp.t1 ──────────────────────────────────────── (12)
/show-jv-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
%0000011 :4V05:$COGNITAS.JV.PERM.ERROR2
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS1
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS2
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS3
%SUM    00005 JV'S; JV-VALUE = 00000011 BYTES
/show-jv-attr jv=# ─────────────────────────────────────────────── (13)
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/show-jv-attr select=*by-attr(password=*write-pass) ────────────── (14)
%0000011 :4V05:$COGNITAS.JV.PERM.ERROR2
%SUM    00001 JV'S; JV-VALUE = 00000011 BYTES
```

(7) The job variable JV.PERM.ERROR2 is protected against unauthorized writing by means of the password 'C5AQ' (the password is not logged).

(8) No new contents can be assigned to the write-protected job variable JV.PERM.ERROR2.

(9) To set new contents, the password had to be entered in the password table of the job.

(10) Output of the new contents 'write error'.

(11) Output of all attributes of the job variable.

(12) The temporary job variable JV.TEMP.1 is created.

(13) Output of all temporary job variables.

(14) Only permanent job variables which are password protected are displayed.

```
/create-jv jv=jv.perm.error3,prot=(basic-acl=*std)─────────────────────── (15)
/show-jv-attr jv=jv.perm.error3,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR3
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% OWNER      = R W         GROUP      = - -          OTHERS     = - -
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:22:23  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/mod-jv-attr jv=jv.perm.error3,new-name=#jv.temp.t3────────────────────── (16)
%  JVS0449 ONLY DEFAULT ATTRIBUTES PERMITTED FOR TEMPORARY JOB VARIABLE.
COMMAND REJECTED
/mod-jv-attr jv=jv.perm.error3,new-name=#jv.temp.t3,prot=(basic-acl=*none)
/show-jv-attr jv=#,inf=*all-attr
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:20:56  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T3
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:22:23  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00002 JV'S; JV-VALUE = 00000000 BYTES
/show-jv-attr jv=#j*
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T3
%SUM    00002 JV'S; JV-VALUE = 00000000 BYTES
```

(15) The permanent job variable JV.PERM.ERROR3 is created and protected by a basic ACL.

(16) Renaming the permanent job variable JV.PERM.ERROR3 as the temporary job variable JV.TEMP.T3 is possible only when the protection attributes are explicitly reset to the default value.

```
/delete-jv jv=#jv.temp. ————————————————————————————————————————————————— (17)
%  JVS0465 DELETE ALL JOB VARIABLES ':4V05:$COGNITAS.S.187.0FDB.JV.TEMP.' OF
USER ID? REPLY (Y=YES; N=NO; T=TERMINATE COMMAND; ?=EXPLAIN ADDITIONAL
OPTIONS)?y
/delete-jv jv=jv.perm.status* ——————————————————————————————————————————— (18)
%  JVS0465 DELETE ALL JOB VARIABLES ':4V05:$COGNITAS.JV.PERM.STATUS*' OF USER
ID? REPLY (Y=YES; N=NO; T=TERMINATE COMMAND; ?=EXPLAIN ADDITIONAL
OPTIONS)?y,check=single
%  JVS0469 DELETE JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.STATUS1'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
%  JVS0469 DELETE JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.STATUS2'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
%  JVS0469 DELETE JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.STATUS3'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
/show-jv-attr jv.perm.,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:11:13  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%0000011 :4V05:$COGNITAS.JV.PERM.ERROR2
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:17:41  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = YES
%SUM    00002 JV'S; JV-VALUE = 00000011 BYTES
/mod-jv-attr jv=jv.perm.error1,prot=(basic-acl=*previous) ——————————————— (19)
/show-jv-attr jv.perm.error1,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% OWNER      = R W        GROUP      = - -         OTHERS     = - -
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:11:13  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
```

(17) All temporary job variables with names beginning with “JV.TEMP.” are to be deleted.

(18) All permanent job variables beginning with JV.PERM.STATUS are to be deleted. During deletion the check mode is changed for JV.PERM.STATUS* job variables: the check query is made for each job variable to be deleted.

(19) The job variable JV.PERM.ERROR1 is protected by a basic ACL, the values for which are set in accordance with the standard access control.

```
/create-jv jv=jv.perm.status5,prot=(basic-acl=*std) ──────────────── (20)
/show-jv-attr jv=**status5,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS5
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% OWNER      = R W         GROUP      = - -         OTHERS     = - -
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:32:25  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/create-jv jv=jv.perm.status6 ────────────────────────────────────── (21)
/mod-jv-attr jv=jv.perm.status6,prot=(retention-period=10)
/show-jv-attr jv=**status6,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS6
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-14
% CRE-TIME   =   14:34:54  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/show-jv-attr select=*by-attr(basic-acl=*yes) ─────────────────────── (22)
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
%0000000 :4V05:$COGNITAS.JV.PERM.STATUS5
SUM    00002 JV'S; JV-VALUE = 00000000 BYTES
/set-jv-link link=status6,jv=jv.perm.status6
/set-jv-link link=stat,jv=jv.perm.status6 ─────────────────────────── (23)
/show-jv-link
% LINK-NAME  JV-NAME
% *STAT      :4V05:$COGNITAS.JV.PERM.STATUS6
% *STATUS6   :4V05:$COGNITAS.JV.PERM.STATUS6
/set-jv-link link=stat,jv=jv.perm.status5 ─────────────────────────── (24)
/show-jv-link
% LINK-NAME  JV-NAME
% *STAT      :4V05:$COGNITAS.JV.PERM.STATUS5
% *STATUS6   :4V05:$COGNITAS.JV.PERM.STATUS6
```

(20) The job variable JV.PERM.STATUS5 is newly created and simultaneously protected by a basic ACL in which only the owner has all access rights .

(21) Only job variables which are protected with a BASIC-ACL are displayed.

(22) The job variable JV.PERM.STATUS6 is newly created and subsequently protected against modification for ten days.

(23) For the job variable JV.PERM.STATUS6, two entries under the link names STATUS6 and STAT are set up in the JV-LINK table for the job.

(24) For the job variable JV.PERM.STATUS5, one entry is set up under the link name STAT, thereby overwriting the existing link name.

```
/set-jv-link link=temp1,jv=#jv.temp.t1 ─────────────────────────────────── (25)
/show-jv-link
% LINK-NAME  JV-NAME
% *STAT      :4V05:$COGNITAS.JV.PERM.STATUS5
% *STATUS6   :4V05:$COGNITAS.JV.PERM.STATUS6
% *TEMP1     :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
/show-jv-attr jv=*link(link=temp1),inf=*all-attr ────────────────────────── (26)
%0000000 :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:38:41  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/del-jv jv=*link(link=status6) ─────────────────────────────────────────── (27)
%  JVS04A3 ERROR WHEN DELETING JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.STATUS6'
%  JVS04B6 EXPIRATION DATE FOR JOB VARIABLE NOT YET REACHED. COMMAND REJECTED
/del-jv jv=*link(link=status6),ignore-prot=*expir ───────────────────────── (28)
/del-jv jv=jv.perm.status5,dialog-control=*jv-change ────────────────────── (29)
%  JVS0469 DELETE JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.STATUS5'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
/show-jv-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
%0000011 :4V05:$COGNITAS.JV.PERM.ERROR2
%SUM    00002 JV'S; JV-VALUE = 00000011 BYTES
/show-jv-link ──────────────────────────────────────────────────────────── (30)
% LINK-NAME  JV-NAME
% *STAT      :4V05:$COGNITAS.JV.PERM.STATUS5
% *STATUS6   :4V05:$COGNITAS.JV.PERM.STATUS6
% *TEMP1     :4V05:$COGNITAS.S.187.0FDB.JV.TEMP.T1
```

(25) For the temporary job variable JV.TEMP.T1, an entry is set up under the link name TEMP1 . Since the job variable does not yet exist, it is newly created by the system.

(26) The job variables can be referenced in commands via the link names.

(27) The job variable JV.PERM.STATUS6 is to be deleted; it is to be referenced via its link name. The job variable cannot be deleted because its expiration date has not yet been reached.

(28) The job variable JV.PERM.STATUS6 can be deleted if the expiration date is not taken into consideration.

(29) A control query is executed for the job variable to be deleted by specifying the operand `dialog-control=*jv-change`. The job variable JV.PERM.STATUS5 is deleted.

(30) Although the job variables JV.PERM.STATUS5 and JV.PERM.STATUS6 were deleted, all JV-LINK entries are still present.

```
/mod-jv-attr jv=jv.perm.error1,prot=(basic-acl=(owner=(read=y,write=n),
      group=*no-access,others=*no-access)) ──────────────────────────── (31)
/show-jv-attr jv.perm.error1,inf=*all-attr
%0000000 :4V05:$COGNITAS.JV.PERM.ERROR1
% USER-ACC   = OWNER-ONLY  ACCESS     = WRITE
% OWNER      = R -         GROUP      = - -         OTHERS     = - -
% CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
% CRE-TIME   =   14:11:13  EXPIR-TIME =   00:00:00
% READ-PASS  = NONE
% WRITE-PASS = NONE
%SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
/del-jv jv=jv.perm.error1,dialog-control=*jv-change
%  JVS0469 DELETE JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.ERROR1'? REPLY
(Y=YES; N=NO; T=TERMINATE; ,CHECK=NEW MODE)?y
%  JVS04A3 ERROR WHEN DELETING JOB VARIABLE ':4V05:$COGNITAS.JV.PERM.ERROR1'
%  JVS04BF REQUESTED ACCESS TO JV NOT PERMITTED DUE TO EXISTING JV
PROTECTION. COMMAND REJECTED
/del-jv jv=jv.perm.error1,ignore-protection=*access ────────────────────── (32)
```

(31) The job variable is protected with a BASIC-ACL against accidental overwriting. The owner has read-only access and all other users have no access.

(32) When deleting the job variable JV.PERM.ERROR1, the protection attributes (here the BASIC-ACL) must be taken into considerations.

Example 2 (program monitoring)

The second example illustrates the use of monitoring job variables for communication between programs. There are two programs, each within a separate task.
Program 1 is to run whatever the circumstances, while the decision whether program 2 runs or not depends upon the outcome of program 1. In other words, program 2 must wait for program 1 to finish and check whether it was terminated normally.
The monitoring job variable is deleted by JOB2 after a check query. Both jobs are started under the same user ID. The job can provide further processing steps after program 1, regardless of the execution of job JOB2.

```
/       SET-LOGON-PARAMETERS JOB-NAME=JOB1
/       START-PROGRAM FROM-FILE=PROGRAM-1,MONJV=JV.PROG1
/       SET-JOB-STEP
/       SKIP-COMMANDS TO-LABEL=FEHL,IF=*JV(CONDITION=((JV.PROG1,1,2)=C'$A'))
/       START-EXE FROM-FILE=PROGRAM-XY 1
                         .
                         .
                         .
                         .
/.FEHL  SHOW-JV JV=JV.PROG1
/       EXIT-JOB


/       SET-LOGON-PARAMETERS JOB-NAME=JOB2
/       SHOW-JV JV=JV.PROG1
/       WAIT-EVENT UNTIL=*JV(CONDITION=(((JV.PROG1,1,2)=C'$T' OR -
/                                 (JV.PROG1,1,2)=C'$A')), -
/                          TIME-LIMIT=3600,TIMEOUT-LABEL=FEHL)
/       SKIP-COMMANDS TO-LABEL=FEHL,IF=JV(CONDITION=((JV.PROG1,1,2)=C'$A'))
/       START-EXE FROM-FILE=PROGRAM-2
/.FEHL  SHOW-JV JV=JV.PROG1
/       DELETE-JV JV=JV.PROG1
/       EXIT-JOB
```

[1] As of BLSSERV V2.3, the START-PROGRAM command is replaced by the START-EXECUTABLE-PROGRAM command

Example 3 (program monitoring)

The third example illustrates the use of the TERM macro, via which a user-defined return code can be entered in the job variable monitoring the program.
In the event of an error, a program PROG1 sets the monitoring job variable to a value greater than 1. The user can then interrogate this value in order to decide whether or not it is practical to have a further program PROG2 executed. It should be noted that the status indicator (3 character positions) is set by the system, with the result that the value updated by the user begins in position 4. The length of this value is 4 characters. If PROG2 is to be started by a different job, then PROG2 must first wait for the end of program PROG1:

```
/WAIT-EVENT UNTIL=*JV(CONDITION=((ERROR,1,2)=C'$T') OR ((ERROR,1,2)=C'$A'),
 TIME-LIMIT=...,TIMEOUT-LABEL=...)
```

Program PROG1

```
PROG1  START
       :
FEHL3  LR R9,R15
       :
       TERM URETCD=(R9) ———————————————————————————————————————————————  (1)
       END
       :

/      SET-LOGON-PARAMETERS JOB-NAME=BEISP3
/      SET-JV-LINK JV-NAME=ERROR
/      START-ASSEMBH
//     COMPILE SOURCE=PROG1,...
//     END
/      START-EXE FROM-FILE=*OMF,MONJV=ERROR [1]
/      SET-JOB-STEP
/      SKIP-COMMANDS TO-LABEL=EXIT,IF=*JV(CONDITION=((ERROR,4,4)>C'0001')
/      START-EXE FROM-FILE=PROG2
/.EXIT EXIT-JOB
```

(1) The URETCD operand in the TERM macro ensures that the contents of the specified register are edited and stored to the fourth decimal place in the program-monitoring job variable after the status indicator, in bytes 4-7 of the value field (see also the “Executive Macros” manual [4]).

[1] As of BLSSERV V2.3, the START-PROGRAM command is replaced by the START-EXECUTABLE-PROGRAM command

Example 4 (job monitoring)

The jobs AJOB, BJOB and CJOB are started and monitored from a control job.

```
/          SET-LOGON-PARAMETERS JOB-NAME=STEUER
/          ENTER-JOB FROM-FILE=AJOB,MONJV=JV.AJOB
/          ENTER-JOB FROM-FILE=BJOB,MONJV=JV.BJOB
/          ENTER-JOB FROM-FILE=CJOB,MONJV=JV.CJOB ———————————————————————  (1)
/          SHOW-JOB-STATUS JOB-ID=*MONJV(JV.AJOB) ———————————————————————  (2)
/          WAIT-EVENT UNTIL=*JV(CONDITION=((JV.AJOB,1,2)=C'$T' OR -
/                                         (JV.AJOB,1,2)=C'$A'),-
/                               TIME-LIMIT=3600,TIMEOUT-LABEL=FEHL1) ———  (3)
/          SKIP-COMMANDS TO-LABEL=NORMAL,IF=*JV(CONDITION=((JV.AJOB,1, -
                                                          2)=C'$T')) —  (4)
/.FEHL1    CANCEL-JOB JOB-ID=*MONJV(JV.CJOB) ———————————————————————————  (5)
/.NORMAL   SET-JOB-STEP
/          SKIP-COMMANDS TO-LABEL=ENDE,IF=*JV(CONDITION=((JV.BJOB,1,2) -
                                                        NE C'$R'))
/          CHANGE-TASK-PRIORITY JOB-ID=*MONJV(JV.BJOB),RUN-PRIORITY=130 —  (6)
/.ENDE     EXIT-JOB
```

(1) The jobs are started and monitored.

(2) Information about AJOB is interrogated.

(3) Wait until AJOB terminated (normally or abnormally), but for a maximum of 3600 seconds.

(4) Check whether AJOB was terminated normally or abnormally.

(5) In the event of abnormal termination of AJOB, processing of CJOB is likewise aborted.

(6) If CJOB is still running, the priority is changed.

Example 5 (job monitoring)

The following interdependencies exist between the jobs AJOB, BJOB, CJOB, EJOB and FJOB: AJOB should not be started until BJOB has been normally terminated.
When CJOB has been normally terminated, EJOB and FJOB should be started simultaneously.

The desired control is implemented by means of the following coordination job:

```
/SET-LOGON-PARAMETERS JOB-NAME=MASTER
/REMARK ******************************************
/REMARK **         START BJOB AND CJOB          **
/REMARK ******************************************
/ENTER-JOB FROM-FILE=BJOB,MONJV=JV.BJOB,JOB-NAME=BJOB
/ENTER-JOB FROM-FILE=CJOB,MONJV=JV.CJOB,JOB-NAME=CJOB
/REMARK ******************************************
/REMARK **  WHEN BJOB FINISHED, START AJOB      **
/REMARK ******************************************
/ADD-CJC-ACTION CONDITION=((JV.BJOB,1,2)=C'$T'),-
/               NAME=BJOB,TIME-LIMIT=3600
/ENTER-JOB FROM-FILE=AJOB,JOB-NAME=AJOB
/END-CJC-ACTION
/REMARK ******************************************
/REMARK ** WHEN CJOB FINISHED, START EJOB AND FJOB*
/REMARK ******************************************
/ADD-CJC-ACTION CONDITION=((JV.CJOB,1,2)=C'$T'),-
/               NAME=CJOB,TIME-LIMIT=3600
/ENTER-JOB FROM-FILE=EJOB,JOB-NAME=EJOB
/ENTER-JOB FROM-FILE=FJOB,JOB-NAME=FJOB
/END-CJC-ACTION
/REMARK ******************************************
/REMARK ** WAIT UNTIL ALL EVENTS HAVE OCCURRED   **
/REMARK ** AND ALL ACTIONS HAVE BEEN STARTED     **
/REMARK ******************************************
/WAIT-EVENT UNTIL=*JV(CONDITION=(((JV.BJOB,1,2)=C'$T' OR -
/                                 (JV.BJOB,1,2)=C'$A') AND -
/                                ((JV.CJOB,1,2)=C'$T' OR -
/                                 (JV.CJOB,1,2)=C'$A')),-
/                     TIME-LIMIT=3600)
/EXIT-JOB
```

Example 6 (job monitoring)

The job SPV checks the execution of JOBA, JOBB and JOBC. By setting the job variable MONA, JOBA influences both the start of JOBB and the “forced” termination of JOBC (CANCEL-JOB). MONB and MONC are monitoring job variables for JOBB and JOBC.

Job SPV

```
/    SET-LOGON-PARAMETERS JOB-NAME=SPV
/    ASSIGN-SYSOUT TO-FILE=OUT.E.SPV
/    REMARK *** DECLARE JOB VARIABLES ***
/    SET-JV-LINK JV-NAME=MONA
/    SET-JV-LINK JV-NAME=MONB
/    SET-JV-LINK JV-NAME=MONC
/    REMARK *** START JOBA ***
/    ENTER-JOB FROM-FILE=JOBA,JOB-CLASS=JCBATCH
/    SHOW-USER-STATUS
/    REMARK *** START JOBB IF MONA SET ***
/    WAIT-EVENT UNTIL=*JV(CONDITION=(MONA=C'START B'),TIME-LIMIT=100)
/    ENTER-JOB FROM-FILE=JOBB,MONJV=MONB,JOB-CLASS=JCBATCH
/    SHOW-USER-STATUS
/    SHOW-JOB-STATUS JOB-ID=*MONJV(MONB)
/    REMARK *** START JOBC IF JOBB TERMINATED ***
/    ADD-CJC-ACTION CONDITION=((MONB,1,2)=C'$T'),TIME-LIMIT=100
/    ENTER-JOB FROM-FILE=JOBC,MONJV=MONC,JOB-CLASS=JCBATCH
/    END-CJC-ACTION
/    WAIT-EVENT UNTIL=*JV(CONDITION=((MONC,1,2)=C'$R'))
/    SHOW-JOB-STATUS JOB-ID=*MONJV(MONC)
/    REMARK *** CANCEL JOBC IF MONA SET ***
/    WAIT-EVENT UNTIL=*JV(CONDITION=(MONA=C'CANCEL C'),TIMEOUT=W1)
/    CANCEL-JOB JOB-ID=*MONJV(MONC)
/.W1 WAIT-EVENT UNTIL=*JV(TIME-LIMIT=10,TIMEOUT-LABEL=W2)
/.W2 SHOW-USER-STATUS
/    WAIT-EVENT UNTIL=*JV(TIME-LIMIT=20,TIMEOUT-LABEL=W3)
/.W3 EXIT-JOB
```

Job JOBA

```
/SET-LOGON-PARAMETERS JOB-NAME=JOBA
/ASSIGN-SYSOUT TO=OUT.E.JOBA
/WRITE-TEXT TEXT=C'***************************************************'
/WRITE-TEXT TEXT=C'***     H E R E  I S   J O B A                 ****'
/WRITE-TEXT TEXT=C'***************************************************'
/MODIFY-JV JV=MONA,SET-VALUE=C'START B'
/WAIT-EVENT UNTIL=*JV(CONDITION=((MONB,1,2)=C'$R'))
/WRITE-TEXT TEXT=C'***************************************************'
/WRITE-TEXT TEXT=C'***     J O B B    S T A R T E D               ****'
/WRITE-TEXT TEXT=C'***************************************************'
/WAIT-EVENT UNTIL=*JV(TIME-LIMIT=60,TIMEOUT-LABEL=W1)
/.W1 MODIFY-JV JV=MONA,SET-VALUE=C'CANCEL C'
/WAIT-EVENT UNTIL=*JV(CONDITION=((MONC,1,2)=C'$A'))
/WRITE-TEXT TEXT=C'***************************************************'
/WRITE-TEXT TEXT=C'***     J O B C    C A N C E L E D             ****'
/WRITE-TEXT TEXT=C'***************************************************'
/EXIT-JOB
```

Job JOBB

```
/SET-LOGON-PARAMETERS JOB-NAME=JOBB
/ASSIGN-SYSOUT TO=OUT.E.JOBB
/WRITE-TEXT TEXT=C'***************************************************'
/WRITE-TEXT TEXT=C'***     H E R E   I S   J O B B                ****'
/WRITE-TEXT TEXT=C'***************************************************'
/WAIT-EVENT UNTIL=*JV(TIME-LIMIT=20,TIMEOUT-LABEL=W1)
/.W1 EXIT-JOB
```

Job JOBC

```
/SET-LOGON-PARAMETERS JOB-NAME=JOBC
/ASSIGN-SYSOUT TO=OUT.E.JOBC
/WRITE-TEXT TEXT=C'***************************************************'
/WRITE-TEXT TEXT=C'***     H E R E   I S   J O B C                ****'
/WRITE-TEXT TEXT=C'***************************************************'
/SET-JV-LINK JV=#LOOP
/.S1 MODIFY-JV JV=#LOOP,SET-VALUE=C'RUN'
/    WAIT-EVENT UNTIL=*JV(CONDITION=(#LOOP=C'END'),-
/                         TIME-LIMIT=20,TIMEOUT-LABEL=S1)
/EXIT-JOB
```

Runtime listing for the checking job SPV (OUT.E.SPV)

```
/    REMARK *** DECLARE JOB VARIABLES ***
/    SET-JV-LINK JV-NAME=MONA
/    SET-JV-LINK JV-NAME=MONB
/    SET-JV-LINK JV-NAME=MONC
/    REMARK *** START JOBA ***
/    ENTER-JOB FROM-FILE=JOBA,JOB-CLASS=JCBATCH
%  JMS0066 JOB 'JOBA' ACCEPTED ON 03-08-04 AT 15:54, TSN = 0FDN
/    SHOW-USER-STATUS
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
JOBA     0FDN 1 WT       9 255      0.0        200 89001
COGDIA   0FDB 3 DIALOG   0 240      0.7721   32767 89001
SPV      0FDM 2 BATCH    9 255      0.0216     200 89001
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
%  SPS0420 RSO WARNING : SOME RSO PRINT-JOBS CANNOT BE DISPLAYED
/    REMARK *** START JOBB IF MONA SET ***
/    WAIT-EVENT UNTIL=*JV(CONDITION=(MONA=C'START B'),TIME-LIMIT=100)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:54:40
%  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 15:54:41
/    ENTER-JOB FROM-FILE=JOBB,MONJV=MONB,JOB-CLASS=JCBATCH
%  JMS0066 JOB 'JOBB' ACCEPTED ON 03-08-04 AT 15:54, TSN = 0FDP
/    SHOW-USER-STATUS
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
JOBB     0FDP 1 WT       9 255      0.0        200 89001
         0FDB 3 DIALOG   0 240      0.7721   32767 89001
SPV      0FDM 2 BATCH    9 255      0.0311     200 89001
JOBA     0FDN 2 BATCH    9 255      0.0150     200 89001
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
%  SRO0376 NO RSO JOB OF TYPE 'T7' PRESENT
%  SPS0420 RSO WARNING : SOME RSO PRINT-JOBS CANNOT BE DISPLAYED
/    SHOW-JOB-STATUS JOB-ID=*MONJV(MONB)
TSN:     0FDP       TYPE:    1 WT       NOW:     2003-08-04.155441
JOBNAME: JOBB       PRI:     9 255      SPOOLIN: 2003-08-04.1554
USERID:  COGNITAS   JCLASS:  JCBATCH    INTYPE:       0
ACCNB:   89001      CPU-MAX:    200     START:   SOON
REPEAT:  NO         RERUN:   NO         FLUSH:   NO
ORIGFILE::4V05:$COGNITAS.JOBB
MONJV:   :4V05:$COGNITAS.MONB
/    REMARK *** START JOBC IF JOBB TERMINATED ***
/    ADD-CJC-ACTION CONDITION=((MONB,1,2)=C'$T'),TIME-LIMIT=100
/    ENTER-JOB FROM-FILE=JOBC,MONJV=MONC,JOB-CLASS=JCBATCH
/    END-CJC-ACTION
%  CJC0050 CJC ACTION WITH ID =    2, LABEL = *NONE: ACCEPTED FOR FURTHER
EVENTS AT 15:54:41, COUNT = 1
/    WAIT-EVENT UNTIL=*JV(CONDITION=((MONC,1,2)=C'$R'))
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:54:41
%  CJC0051 CJC ACTION WITH ID =    2, LABEL = *NONE: CONDITION TRUE AT
15:55:04, COUNT = 0
%  CJC0064 CJC ACTION WITH ID =    2, LABEL = *NONE: START OF ON OR TIMEOUT
SEQUENCE
%  JMS0066 JOB 'JOBC' ACCEPTED ON 03-08-04 AT 15:55, TSN = 0FDR
%  CJC0065 CJC ACTION WITH ID =    2, LABEL = *NONE: END OF ON OR TIMEOUT
SEQUENCE
%  CJC0052 CJC ACTION WITH ID =    2, LABEL = *NONE: TERMINATION NORMAL
%  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 15:55:04
/    SHOW-JOB-STATUS JOB-ID=*MONJV(MONC)
TSN:     0FDR       TYPE:    2 BATCH    NOW:     2003-08-04.155504
JOBNAME: JOBC       PRI:     9 255      SPOOLIN: 2003-08-04.1555
USERID:  COGNITAS   JCLASS:  JCBATCH    LOGON:   2003-08-04.1555
ACCNB:   89001      CPU-MAX:    200     CPU-USED:000000.0050
REPEAT:  NO         RERUN:   NO         FLUSH:   NO
MRSCAT:             HOLD:    NO         START:   SOON
TID:     00010067   UNP/Q#:    00/000
CMD:
ORIGFILE::4V05:$COGNITAS.JOBC
MONJV:   :4V05:$COGNITAS.MONC
/    REMARK *** CANCEL JOBC IF MONA SET ***
/   WAIT-EVENT UNTIL=*JV(CONDITION=(MONA=C'C CANCELN'),TIMEOUT-LABEL=W1)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:04
%  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 15:55:41
/   CANCEL-JOB JOB-ID=*MONJV(MONC)
%  CAN000K CANCEL PROCESSING STARTED FOR TSN '0FDR' WITH USER ID 'COGNITAS'
/.W1 WAIT-EVENT UNTIL=*JV(TIME-LIMIT=10,TIMEOUT-LABEL=W2)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:41
%  CJC0022 WAIT COMMAND: TIMEOUT AT 15:55:52, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.W2 SHOW-USER-STATUS
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
         0FDB 3 DIALOG   0 240      0.7756   32767 89001
SPV      0FDM 2 BATCH    9 255      0.0525     200 89001
JOBB     0FDQ 4 PR  255      1       0      0   0FDP
JOBC     0FDS 4 PR  255      1       0      0   0FDR
JOBA     0FDT 4 PR  255      1       0      0   0FDN
%  SPS0420 RSO WARNING : SOME RSO PRINT-JOBS CANNOT BE DISPLAYED
/    WAIT-EVENT UNTIL=*JV(TIME-LIMIT=20,TIMEOUT=W3)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:52
%  CJC0022 WAIT COMMAND: TIMEOUT AT 15:56:12, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.W3 EXIT-JOB
%  EXC0419 /LOGOFF AT 1556 ON 03-08-04 FOR TSN '0FDM'
%  EXC0421 CPU TIME USED: 0.0595
```

Runtime listing for job JOBA (OUT.E.JOBA)

```
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WRITE-TEXT TEXT=C'***     H E R E  I S   J O B A                 ****'
****     H E R E  I S   J O B A                 ****
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/MODIFY-JV JV=MONA,SET-VALUE=C'START B'
/WAIT-EVENT UNTIL=*JV(CONDITION=((MONB,1,2)=C'$R'))
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:54:40
%  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 15:54:41
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WRITE-TEXT TEXT=C'***     J O B B    S T A R T E D               ****'
***     J O B B    S T A R T E D               ****
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WAIT-EVENT UNTIL=*JV(TIME-LIMIT=60,TIMEOUT-LABEL=W1)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:54:41
%  CJC0022 WAIT COMMAND: TIMEOUT AT 15:55:41, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.W1 MODIFY-JV JV=MONA,SET-VALUE=C'CANCEL C'
/WAIT-EVENT UNTIL=*JV(CONDITION=((MONC,1,2)=C'$A'))
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:41
%  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 15:55:42
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WRITE-TEXT TEXT=C'***     J O B C    C A N C E L E D             ****'
***     J O B C    C A N C E L E D             ****
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/EXIT-JOB
%  EXC0419 /LOGOFF AT 1555 ON 03-08-04 FOR TSN 'OFDN'
%  EXC0421 CPU TIME USED: 0.0257
```

Runtime listing for job JOBB (OUT.E.JOBB)

```
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WRITE-TEXT TEXT=C'***     H E R E   I S   J O B B                ****'
***     H E R E   I S   J O B B                ****
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WAIT-EVENT UNTIL=*JV(TIME-LIMIT=20,TIMEOUT-LABEL=W1)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:54:41
%  CJC0022 WAIT COMMAND: TIMEOUT AT 15:55:02, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.W1 EXIT-JOB
%  EXC0419 /LOGOFF AT 1555 ON 03-08-04 FOR TSN 'OFDP'
%  EXC0421 CPU TIME USED: 0.0178
```

Runtime listing for job JOBC (OUT.E.JOBC)

```
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/WRITE-TEXT TEXT=C'***     H E R E   I S   J O B C                ****'
***     H E R E   I S   J O B C                ****
/WRITE-TEXT TEXT=C'***************************************************'
***************************************************
/SET-JV-LINK JV=#LOOP
/.S1 MODIFY-JV JV=#LOOP,SET-VALUE=C'RUN'
/    WAIT-EVENT UNTIL=*JV(CONDITION=(#LOOP=C'END'),
TIME-LIMIT=20,TIMEOUT-LABEL=S1)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:04
%  CJC0022 WAIT COMMAND: TIMEOUT AT 15:55:25, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.S1 MODIFY-JV JV=#LOOP,SET-VALUE=C'RUN'
/    WAIT-EVENT UNTIL=*JV(CONDITION=(#LOOP=C'END'),
TIME-LIMIT=20,TIMEOUT-LABEL=S1)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 15:55:25
%  CAN00BY CANCELLED BY 'BTCH OFDM COGNITAS SPV'
%  NRTT201 TASK TERMINATION DUE TO /CANCEL(-JOB)  COMMAND
%  EXC0419 /LOGOFF AT 1555 ON 03-08-04 FOR TSN 'OFDR'
%  EXC0421 CPU TIME USED: 0.0245
```

Example 7 (job variable macros, without ONEVT macro)

The following sample program SRC.BJV illustrates the use of certain JV macros:

Source program SRC.BJV

```
BJV      START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         STAJV JV.A,STA,120,OUTPUT=NEW,VERSION=4 ----------------------  (1)
DTH1     DCLJV JV.A,LINK=*LINK,VERSION=1 ------------------------------  (2)
         MVC   STA,NULL2
         STAJV JV.,STA,120,OUTPUT=NEW,VERSION=4 -----------------------  (3)
DTH2     CATJV JV.A,JV.N,STATE=U,RDPASS=C'JV',VERSION=1
         SETJV JV.N,SET1,PASS=C'JV',VERSION=1
         GETJV JV.N,GET,30,PASS=C'JV',VERSION=1
DTH3     MVC   GET,NULL1
         SETJV (JV.N,3,4),SET2,PASS=C'JV',VERSION=1 -------------------  (4)
         GETJV JV.N,GET,30,PASS=C'JV',VERSION=1
DTH4     MVC   GET,NULL1
         SETJV JV.N,SET2,PASS=C'JV',VERSION=1 -------------------------  (5)
         GETJV JV.N,GET,30,PASS=C'JV',VERSION=1
DTH5     MVC   GET,NULL1
         ERAJV JV.N,VERSION=1 -----------------------------------------  (6)
DTH6     MVC   STA,NULL2
         STAJV JV.,STA,120,OUTPUT=NEW,SELADDR=SEL1,VERSION=4 ----------  (7)
DTH7     ERAJV JV.N,PASS=C'JV',VERSION=1 ------------------------------  (8)
*
DTH8     TERM
************************************************
         DS    0F
STA      DS    L120
*
SEL1     JVSEL MF=L,PASS=RDPASS
*
GET      DS    L50
*
SET1     DC    Y(SETEND1-SET1)
         DS    CL2
         DC    'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
SETEND1  EQU   *
*
SET2     DC    Y(SETEND2-SET2)
         DS    CL2
         DC    '34567'
SETEND2  EQU   *
*
```

```
NULL1    DS    0CL50
NULL2    DC    120X'00'
         END
         END
```

Points (1) to (8) in the source program are explained in the following runtime listing.

Runtime listing

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '01.2C00' OF '2002-03-06' LOADED
%  BLS0552 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS GMBH 2010. ALL RIGHTS
RESERVED
%  ASS6010 V01.2C00 OF BS2000 ASSEMBH  READY
//compile source=src.bjv,module-lib=ass.plamlib(elem=bjv),test-support=yes
%  ASS6011 ASSEMBLY TIME: 430 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 59 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-exe from=(lib=ass.plamlib,elem=bjv),test-opt=*aid  [1]
%  BLS0517 MODULE 'BJV' LOADED
/%insert dth1
/resume-program
STOPPED AT LABEL: DTH1 , SRC_REF: 40, SOURCE: BJV , PROC: BJV
/%display %15  ———————————————————————————————————————————————————  (1)
*** TID: 0001005C *** TSN: 0FFF *********************************************
CURRENT PC: 0000009C    CSECT: BJV*******************************************
%15            = 00000433
```

(1) The STAJV macro is to transfer the status of job variable JV.A to field STA. Register 15 shows: JV.A is not in the system.

[1] As of BLSSERV V2.3, the LOAD-PROGRAM command is replaced by the LOAD-EXECUTABLE-PROGRAM command

```
/%insert dth2
/resume-program
STOPPED AT LABEL: DTH2 , SRC_REF: 92, SOURCE: BJV , PROC: BJV
/%display sta
SRC_REF:    92 SOURCE: BJV  PROC: BJV
***************************************
STA            =
|.JV.A.JV.DO.1.JV.E.1.JV.MON.JV.N.JV.PERM.ERROR.JV.PERM.ERROR.READ.JV.PERM.ER
R|
|OR2.JV.PERM.WAIT.JV.PROG...................|
/%display sta%x  ────────────────────────────────────────────────────────  (2)
CURRENT PC: 000001B4    CSECT: BJV
******************************************
V'00000868' = STA      + #'00000000'
00000868 (00000000) 05D1E54B C108D1E5 4BC4D64B F107D1E5    .JV.A.JV.DO.1.JV
00000878 (00000010) 4BC54BF1 07D1E54B D4D6D505 D1E54BD5    .E.1.JV.MON.JV.N
00000888 (00000020) 0ED1E54B D7C5D9D4 4BC5D9D9 D6D913D1    .JV.PERM.ERROR.J
00000898 (00000030) E54BD7C5 D9D44BC5 D9D9D6D9 4BD9C5C1    V.PERM.ERROR.REA
000008A8 (00000040) C40FD1E5 4BD7C5D9 D44BC5D9 D9D6D9F2    D.JV.PERM.ERROR2
000008B8 (00000050) 0DD1E54B D7C5D9D4 4BE6C1C9 E308D1E5    .JV.PERM.WAIT.JV
000008C8 (00000060) 4BD7D9D6 C7000000 00000000 00000000    .PROG...........
000008D8 (00000070) 00000000 00000000                      ........

/%insert dth3
/resume-program
STOPPED AT LABEL: DTH3 , SRC_REF: 188, SOURCE: BJV , PROC: BJV
/%display get  ──────────────────────────────────────────────────────────  (3)
SRC_REF:   188 SOURCE: BJV  PROC: BJV
***************************************
GET            = |..  ABCDEFGHIJKLMNOPQRSTUVWXYZ...................|
```

(2) The DCLJV macro catalogs job variable JV.A and assigns the link name *LINK to it. The STAJV macro transfers the names of all job variables beginning with “JV.” in field STA. The default value LIST=OLD applies, i.e. the output does not list the catalog ID or user ID.

(3) JV.A is renamed JV.N and is given the read password “JV”. The user must specify this password in order to access the job variable.
The SETJV macro sets JV.N to the value specified in field SET1. The GETJV macro then reads this value into the GET field.

```
/%insert dth4
/resume-program
STOPPED AT LABEL: DTH4 , SRC_REF: 258, SOURCE: BJV , PROC: BJV
/%display get  ——————————————————————————————————————————————————————  (4)
SRC_REF:   258 SOURCE: BJV  PROC: BJV
****************************************
GET             = |..  AB3456GHIJKLMNOPQRSTUVWXYZ....................|

/%insert dth5
/resume-program
STOPPED AT LABEL: DTH5 , SRC_REF: 328, SOURCE: BJV , PROC: BJV
/%display get  ——————————————————————————————————————————————————————  (5)
SRC_REF:   328 SOURCE: BJV  PROC: BJV
****************************************
GET             = |..  34567.........................................|

/%insert dth6
/resume-program
STOPPED AT LABEL: DTH6 , SRC_REF: 360, SOURCE: BJV , PROC: BJV
/%display %15  ——————————————————————————————————————————————————————  (6)
CURRENT PC: 000006E6    CSECT: BJV
*******************************************
%15             = 000004B1
```

(4) Starting at byte 3, four bytes of the job variable value are overwritten with the contents of field SET2. The GET field shows the job variable value subsequent to execution of the SETJV macro.

(5) The job variable value is replaced completely, from the start position (=1, default value), by the contents specified in SET2.

(6) The job variable is to be deleted. Since the required read password has not been specified, register 15 contains the appropriate error code.

```
/%insert dth7
/resume-program

STOPPED AT LABEL: DTH7 , SRC_REF: 396, SOURCE: BJV , PROC: BJV
/%display sta
SRC_REF:   396 SOURCE: BJV  PROC: BJV
*************************************
STA            =
|.JV.N..................................................................
.|
|........................................|
/%display sta%x  ————————————————————————————————————————————————————  (7)
CURRENT PC: 00000798    CSECT: BJV
****************************************
V'00000868' = STA      + #'00000000'
00000868 (00000000) 05D1E54B D5000000 00000000 00000000   .JV.N...........
00000878 (00000010) 00000000 00000000 00000000 00000000   ................
          REPEATED LINES:     4
000008C8 (00000060) 00000000 00000000 00000000 00000000   ................
000008D8 (00000070) 00000000 00000000                     ........

/%insert dth8
/resume-program
STOPPED AT LABEL: DTH8 , SRC_REF: 423, SOURCE: BJV , PROC: BJV
/%display %15  ——————————————————————————————————————————————————————  (8)
CURRENT PC: 0000084A    CSECT: BJV
****************************************
%15            = 00000000
```

(7) The STAJV macro gives the names of all job variables beginning with “JV.” and protected with a read password (selection is made with the operand list generated by the JVSEL macro), in the STA field. The default value is LIST=OLD, i.e. the output is made without catid and userid.

(8) The read password is specified for the delete operation and the job variable is successfully deleted.

Example 8 (with ONEVT macro)

The BONEVT program issues an ONEVT macro for the event item ONEVTEST, which causes a POSSIG call whenever one of the following conditions is satisfied:

– job variable JV1 is set to 'MELDUNG' (= “message”) at the time of the ONEVT macro call
– JV1 is set to 'MELDUNG'
– catalog is exported

The number of POSSIG calls is limited to three (COUNT=3), i.e. the ONEVT macro registers up to 3 'condition satisfied' states. SOLSIG solicits a signal and subsequently the post code is checked. Depending on the condition result, the program issues a message (MELD1, MELD2 or FMELD) after every SOLSIG.

Program BONEVT

```
BONEVT   START
         BALR  3,0
         USING *,3
         PRINT NOGEN
ANF      DCLJV JV1,VERSION=1           "Define JV1"
         ENAEI EINAME=ONEVTEST,EIIDRET=KUKE,SCOPE=GLOBAL
         ONEVT 'JV1=''MELDUNG''',EIID=KUKE,POST='B1',COUNT=3,VERSION=1
CHECK    CHKEI EIID=KUKE               "Check queue"
         LR    5,1
         CMD   '%DISPLAY %15,%5'
SIGNAL   MVC   EMPF,NULL               "Solicit signal"
         SOLSIG EIID=KUKE,COND=UNCOND,RPOSTAD=EMPF,LIFETIM=10
*** CHECK POST CODE *****************************************
PRUEF    CLI   ANZ,X'14'               "Event flag"
         BNE   FMELD
         CLC   ONKEN,POST              "POST value"
         BNE   FMELD
         CLI   ERG,X'00'               "JV1 set"
         BE    MELD1
         CLI   ERG,X'08'               "Catalog exported"
         BE    MELD2
         B     FMELD
*** MESSAGES OUTPUT *****************************************
MELD1    WROUT MLDG1,ENDE              "JV1 set"
         B     CHECK
MELD2    WROUT MLDG2,ENDE              "Catalog exported"
         B     FRAGE
FMELD    WROUT FEHLER,ENDE      "No POSSIG received by ONEVT"
         B     FRAGE
FRAGE    WRTRD ABFRAG,,EINB,,5,ENDE    "Terminate program?"
         CLI   ANTW,'N'
         BE    DONEVT                  "Do not terminate program"
         CLI   ANTW,'Y'
         BE    ENDE                     "Terminate"
         B     FRAGE                    "Repeat query"
DONEVT   DONEVT EIID=KUKE,VERSION=1
         DISEI EIID=KUKE                "Delete event item"
         ERAJV JV1,VERSION=1
         B     ANF
ENDE     TERM

*** DEFINITIONS *********************************************
EMPF     DS    0F
ANZ      DS    CL1
ERG      DS    CL1
ONKEN    DS    CL2
POST     DC    C'B1'
KUKE     DC    C'KURZ'
NULL     DC    F'0'
MLDG1    DC    Y(END1-MLDG1)
         DS    L2
         DC    X'01'
         DC    'JV1 WAS SET'
END1     EQU   *
MLDG2    DC    Y(END2-MLDG2)
         DS    L2
         DC    X'01'
         DC    'CATALOG WAS EXPORTED'
END2     EQU   *
FEHLER   DC    Y(FEND-FEHLER)
         DS    L2
         DC    X'01'
         DC    '***   ERROR   ***'
FEND     EQU   *
ABFRAG   DC    Y(ABEND-ABFRAG)
         DS    CL2
         DC    X'01'
          DC    'TERMINATE PROGRAM?(Y/N)'

ABEND    EQU   *
EINB     DS    0CL5
         DS    CL4
ANTW     DS    CL1
          END
```

The effect of the ONEVT macro is demonstrated in this example using the condition “JV1 is set to 'MELDUNG' ”:

Runtime listing LST.BONEVT for the BONEVT program

```
(IN)     mod-job-options logging=*par(listing=*yes)
(IN)     delete-system-file system-file=*omf
(IN)     start-assembh ─────────────────────────────────────────────────  (1)
(OUT)    % BLS0500 PROGRAM 'ASSEMBH',VERSION '01.2C00' OF '2002-03-06' LOADED
(OUT)    %  BLS0552 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS GMBH 2010.
(  )        ALL RIGHTS RESERVED
(OUT)    %  ASS6010 V01.2C00 OF BS2000 ASSEMBH  READY
(IN)     compile source=bonevt,mod-lib=ass.plamlib(bonevt),test-support=yes
(OUT)    ASS6011 ASSEMBLY TIME: 604 MSEC
(OUT)    %  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
(OUT)    %  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
(OUT)    %  ASS6006 LISTING GENERATOR TIME: 66 MSEC
(IN)     end
(OUT)    %  ASS6012 END OF ASSEMBH
(IN)     load-exe from=(lib=ass.plamlib,elem=bonevt),test-opt=*aid ─────  (2)
(OUT)    %  BLS0517 MODULE 'BONEVT' LOADED
(IN)     %insert check ─────────────────────────────────────────────────  (3)

(IN)     resume-program
(OUT)
(OUT)   STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     mod-jv jv=jv1,set-val='MELDUNG' ───────────────────────────────  (4)
(IN)     mod-jv jv=jv1,set-val='MELDUNG'
(IN)     mod-jv jv=jv1,set-val='MELDUNG'
(IN)     mod-jv jv=jv1,set-val='MELDUNG'
(IN)     mod-jv jv=jv1,set-val='MELDUNG'

(IN)     resume-program
(OUT)    *** TID: 0001005E *** TSN: 0FDB ***********************************
(NL)     CURRENT PC: 000000E0    CSECT: BONEVT  ****************************
(NL)     %15            = 2C000000 ─────────────────────────────────────  (5)
(NL)     %5             = 00000003 ─────────────────────────────────────  (6)
```

(1) The BONEVT program is assembled by the ASSEMBH Assembler and is put in the ASS.PLAMLIB library.

(2) The program is loaded; it is checked using AID (Advanced Interactive Debugger).

(3) The test point CHECK is defined using the AID command %INSERT.

(4) The job variable JV1 is set to 'MELDUNG' five times.

(5) The CHKEI macro shows the POSSIG queue (register 15).

(6) The CHKEI macro shows that the number of POSSIG calls is 3 (register 5).

```
(OUT)    JV1 WAS SET ————————————————————————————————————————————————————  (7)
(OUT)
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT

(IN)     %display EMPF%X ————————————————————————————————————————————————  (8)
(OUT)    CURRENT PC: 000000A8    CSECT: BONEVT  ****************************
(NL)     V'0000028C' = EMPF     + #'00000000'
(NL)     0000028C (00000000) 1400C2F1                             ..B1


(IN)     resume-program
(OUT)    CURRENT PC: 000000E0    CSECT: BONEVT  ****************************
(NL)     %15            = 2C000000
(NL)     %5             = 00000002
(OUT)    JV1 WAS SET
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     resume-program
(OUT)    %15            = 2C000000
(NL)     %5             = 00000001
(OUT)    JV1 WAS SET
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     resume-program
(OUT)    %15            = 30000000
(NL)     %5             = 8F0000AC
(OUT)    ***   ERROR   *** ——————————————————————————————————————————————  (9)

(OUT)   TERMINATE PROGRAM?(Y/N)
(IN)     N
(OUT)
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
```

(7) The first POSSIG was requested with SOLSIG. The condition result is interrogated and an appropriate message output.

(8) After the SOLSIG call, the target field EMPF contains the passed post code:

X'14'	POSSIG was caused by an ONEVT macro.
X'00'	Conditional event “Job variable set”.
X'C2F1'	ONVEVT identifier B1.

(9) After all 3 POSSIG calls have been solicited by SOLSIG, the queue is empty (register 15 of the CHKEI macro: SI=X'30'). The post code check following a further SOLSIG produces the message “ERROR”. The response “N” is given after the prompt, i.e. the program is run one more time starting from the ANF label - a new ONEVT macro is issued.

```
(IN)     mod-jv jv=jv1,set-val='MELDUNG' ──────────────────────────────  (10)
(IN)     mod-jv jv=jv1,set-val='MELDUNG'

(IN)     resume-program
(OUT)    %15             = 2C000000
(NL)     %5             = 00000002 ───────────────────────────────────  (11)
(OUT)    JV1 WAS SET
(OUT)
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     del-jv jv=jv1 ────────────────────────────────────────────────  (12)
(OUT)    %  JVS04A3 ERROR WHEN DELETING JOB VARIABLE ':4V05:$COGNITAS.JV1'
(OUT)    %  JVS0447 JV NAME BEING USED BY CJC FUNCTION. COMMAND REJECTED

(IN)     resume-program
(OUT)    %15             = 2C000000
(NL)     %5              = 00000001
(OUT)    JV1 WAS SET
(OUT)
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     resume-program
(OUT)    %15             = 30000000
(NL)     %5              = 8F0000AC
(OUT)    ***   ERROR   ***
(OUT)    TERMINATE PROGRAM?(Y/N)
(IN)     N
(OUT)
(OUT)    STOPPED AT LABEL: CHECK ,SRC_REF: 84, SOURCE: BONEVT, PROC: BONEVT
(IN)     show-jv-attr jv=jv1 ──────────────────────────────────────────  (13)
(OUT)    0000000 :4V05:$COGNITAS.JV1
(NL)     SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
(IN)     show-jv-attr jv=jv1,inf=*all-attr
(OUT)    000000 :4V05:$COGNITAS.JV1
(NL)      USER-ACC   = ALL-USERS   ACCESS     = WRITE
(NL)      CRE-DATE   = 2003-08-04  EXPIR-DATE = 2003-08-04
(NL)      CRE-TIME   =   16:51:15  EXPIR-TIME =   00:00:00
(NL)      READ-PASS  = NONE
(NL)      WRITE-PASS = NONE
(NL)     SUM    00001 JV'S; JV-VALUE = 00000000 BYTES
```

(10) The “Set JV1” event is signaled twice.

(11) The number of POSSIG calls is 2.

(12) The attempt to delete job variable JV1 is rejected because job variables used in macros (and commands) are protected against deletion.

(13) The attributes of job variable JV1 are output.

```
(IN)     show-jv jv=jv1 ─────────────────────────────────────────────── (14)
(OUT)    %  JVS04B2 SPECIFIED JOB VARIABLE SUBSTRING EMPTY OR ILLEGAL.
(   )    COMMAND REJECTED
(IN)     resume-program
(OUT)    %15           = 30000000
(NL)     %5            = 8F0000AC
(OUT)    ***   ERROR   ***
(OUT)    TERMINATE PROGRAM?(Y/N)
(IN)     Y
(IN)     del-jv jv=jv1 ──────────────────────────────────────────────── (15)
(IN)     show-j-attr jv=jv1
(OUT)    %  JVS0433 REQUESTED JOB VARIABLE NOT CATALOGED. COMMAND REJECTED
(IN)     assign-syslst to=*prim
```

(14) No value is assigned to job variable JV1.

(15) The job variable JV1 can be deleted.

Example 9

This example demonstrates the use of the MODIFY-JV-CONDITIONALLY command and the CSWJV, LNKJV and TIMJV macros. Information is exchanged between a procedure and an ENTER job.

The sample procedure is started by means of the command `/CALL-PROCEDURE PROC.BSP9`. The executable module TESTJV must be stored in the library ASS.PLAMLIB.

Procedure PROC.BSP9

```
/        BEG-PROC LOGGING=*CMD
/        ASSIGN-SYSLST TO=LST.PROC
/        MODIFY-JOB-OPTION LOGGING=*PARAMETER(LISTING=*YES)
/        SET-JV-LINK JV-NAME=JV.E.1
/        MODIFY-JV   JV=JV.E.1,SET-VALUE=C'***'
/        SET-JV-LINK JV-NAME=JV.DO.1
/        MODIFY-JV   JV=JV.DO.1,SET-VALUE=C'***'
/        SET-JV-LINK JV-NAME=JV.MON
/        ENTER-JOB FROM-FILE=TEST.ENTER,MONJV=JV.MON,JOB-CLASS=JCBATCH
/.SKIP1  SHOW-JV JV=JV.MON
/        WAIT-EVENT *JV(CONDITION=((JV.MON,1,2)=C'$R'), -
/                   TIME-LIMIT=90,TIMEOUT-LABEL=SKIP1)
/.PRUEF  MODIFY-JV-CONDITIONALLY JV=(JV.DO.1),-
/                     IF-VALUE=C'GESTARTET',SET-VALUE=C'UNDERSTOOD',-
/                     LABEL=WEIT1
/        SKIP-COMMANDS TO-LABEL=PRUEF
/.WEIT1  SHOW-JV JV=JV.DO.1
/        SHOW-JV JV=JV.E.1
/        SHOW-JV JV=JV.MON
/        SHOW-JOB-STAT *MONJV(JV.MON)
/        WAIT-EVENT *JV(CONDITION=((JV.DO.1)=C'CARRY ON'), -
/                   TIME-LIMIT=15,TIMEOUT-LABEL=WEIT1)
/.WEIT2  MODIFY-JV JV=JV.E.1,SET-VALUE=C'PROGRAM CONTINUE'
/.SKIP3  SKIP-COMMANDS TO-LABEL=ENDE,-
/                     IF=*JV(CONDITION=((JV.E.1,12,3)=C'END'))
/        SHOW-USER-STATUS
/        SHOW-JV JV=JV.E.1
/        SHOW-JV JV=JV.MON
/        WAIT-EVENT UNTIL=*JV(TIME-LIMIT=45,TIMEOUT-LABEL=SKIP3)
/.ENDE   REMARK **** TESTJOB ENDED ****
/        SHOW-JV JV=JV.MON
/        ASSIGN-SYSLST TO=*PRIMARY
/        MODIFY-JOB-OPTION
/        END-PROCEDURE
```

```
/        BEG-PROC LOGGING=*CMD
/        ASSIGN-SYSLST TO=LST.PROC
/        MODIFY-JOB-OPTION LOGGING=*PARAMETER(LISTING=*YES)
```

File TEST.ENTER

```
/.TESTJV SET-LOGON-PARAMETERS
/        ASSIGN-SYSOUT TO=OUT.E.TESTJV
/        SET-JV-LINK JV-NAME=JV.DO.1
/        SET-JV-LINK JV-NAME=JV.PROG
/        MODIFY-JV JV=JV.DO.1,SET-VALUE=C'GESTARTET'
/        START-EXE FROM-FILE=(LIB=ALF.ASS.PLAMLIB,ELEM=TESTJV),-
/                   TEST-OPTIONS=*AID,MONJV=JV.PROG
/        SHOW-JV JV=JV.MON
/        SHOW-JV JV=JV.PROG
/        SKIP-COMMANDS TO-LABEL=ENDE
/        SET-JOB-STEP
/        MODIFY-JV JV=JV.E.1,SET-VALUE=C'ABNORMAL END'
/.ENDE   WAIT-EVENT UNTIL=*JV(TIME-LIMIT=120,TIMEOUT-LABEL=TIME)
/.TIME   MODIFY-JV JV=(JV.E.1,12,3),SET-VALUE=C'END'
/        EXIT-JOB SYSTEM-OUTPUT=*NONE
```

Source program SRC.TESTJV

```
TESTJV   START
         TITLE 'TEST PROGRAM FOR CSWJV'
*
* REGISTER ASSIGNMENT
         BALR  3,0
         USING *,3
         PRINT NOGEN
*
* JOBVARIABLE JV.E.1 IS ASSIGNED THE LINK NAME *ENTER
* LINK NAME *ENTER IS STORED IN THE LNKBER FIELD
* THE CMD MACRO CAUSES THE CONTENTS OF LNKBER TO BE OUTPUT
*
         DCLJV JV.E.1,LINK=*ENTER,VERSION=1
         LNKJV LNKBER,LINK=*ENTER,VERSION=1
         CMD   '%DISPLAY','LNKBER'
*
* THE VALUE OF JV.DO.1 IS COMPARED WITH THE CONTENTS OF VERGL
* IF THEY ARE EQUAL THE SET VALUE NEUWERT IS TO BE SET
*
CSWJV    CSWJV JV.DO.1,VERGL,NEUWERT,VERSION=1
*
* MESSAGE MELD1 *** CSWJV EXECUTED *** IS OUTPUT
*     AND MESSAGE AND TIMESTAMP ARE SET IN MONJV
*
         WROUT MELD1,TERM
         TIMJV MF=S,MONJV=*SMONJVJ,TIMESTAMP=*SET,DESCRIPTOR='TESTJV', -
               INFO='*** CSWJV EXECUTED ***'
         CMD   'SHOW-JV','JV=*LINK(SMONJVJ)'
```

```
*
* THE JVDOW FIELD IS SUPPLIED WITH THE CONTENTS OF JV.DO.1
* COMPARISON TO TEST IF JVDOW CONTAINS THE VALUE 'CARRY ON'
*
         GETJV JV.DO.1,JVDOW,30,VERSION=1
         CLC   JVDOW+4(12),='CARRY ON'
         BE    FORTSETZ
*
* AS THE VALUE OF JV.DO.1 IS MOVED TO THE VERGL FIELD IF THE VALUES
* ARE NOT EQUAL, VERGL IS RESET TO 'UNDERSTOOD'
*
         MVC   VERGLF,='UNDERSTOOD'
         VPASS 1
         B     CSWJV
*
* THE JVEW FIELD IS SUPPLIED WITH THE VALUE OF JV.E.1
* COMPARISON TO TEST IF JVEW CONTAINS THE VALUE 'CONTINUE PROGRAM'
*
FORTSETZ EQU   *
         GETJV JV.E.1,JVEW,30,VERSION=1
         CLC   JVEW+4(19),='CONTINUE PROGRAM'
         BNE   FORTSETZ
*
       MESSAGE *** PROGRAM CONTINUED *** IS OUTPUT
* AND MESSAGE AND TIMESTAMP ARE SET IN MONJV
*
         WROUT MELD2,TERM
         TIMJV MF=S,MONJV=*SMONJVJ,TIMESTAMP=*SET,DESCRIPTOR='TESTJV', -
               INFO=C'*** PROGRAM CONTINUED ***'
         CMD   'SHOW-JV','JV=*LINK(SMONJVJ)'
*
*      MESSAGE *** PROGRAM FINAL SPURT *** IS OUTPUT
* AND MESSAGE AND TIMESTAMP ARE SET IN MONJV
*
BEENDEN  EQU   *
         WROUT MELD3,TERM
         TIMJV MF=S,MONJV=*SMONJVJ,TIMESTAMP=*SET,DESCRIPTOR='TESTJV', -
               INFO=C'*** PROGRAM FINAL SPURT ***'
         CMD   'SHOW-JV','JV=*LINK(SMONJVJ)'
*
* THE VALUE 'NORMAL   END' IS ASSIGNED TO JOB VARIABLE JV.E.1
* AND MESSAGE AND TIMESTAMP ARE SET IN MONJV
*
         SETJV JV.E.1,EWERT,VERSION=1
         TIMJV MF=S,MONJV=*SMONJVJ,TIMESTAMP=*SET,DESCRIPTOR='TESTJV', -
               INFO=C'*** PROGRAM ENDED ***'
TERM     TERM
```

```
*
*        DEFINITIONS
*
VERGL    DC    Y(END1-VERGL)
         DS    CL2
VERGLF   DC    'UNDERSTOOD'
END1     EQU   *
*
NEUWERT  DC    Y(END2-NEUWERT)
         DS    CL2
         DC    'CARRY ON'
END2     EQU   *
*
EWERT    DC    Y(END3-EWERT)
         DS    CL2
         DC    'NORMAL   END'
END3     EQU   *
*
MELD1    DC    Y(END4-MELD1)
         DS    CL2
         DC    X'01'
MELD1TXT DC    '*** CSWJV EXECUTED ***'
END4     EQU   *
*
MELD2    DC    Y(END5-MELD2)
         DS    CL2
         DC    X'01'
MELD2TXT DC    '*** PROGRAM CONTINUED ***'
END5     EQU   *
*
*MELD3    DC    Y(END6-MELD3)
         DS    CL2
         DC    X'01'
MELD3TXT DC    '*** PROGRAM FINAL SPURT ***'
END6     EQU   *
*
JVDOW    DS    CL30
*
JVEW     DS    CL30
*
LNKBER   DS    CL63
         END
```

Runtime listing LST.PROC for procedure PROC.BSP9

```
(IN)     /        MODIFY-JOB-OPTION LOGGING=*PARAMETER(LISTING=*YES)
(IN)     /        SET-JV-LINK JV-NAME=JV.E.1
(IN)     /        MODIFY-JV   JV=JV.E.1,SET-VALUE=C'***'
(IN)     /        SET-JV-LINK JV-NAME=JV.DO.1
(IN)     /        MODIFY-JV   JV=JV.DO.1,SET-VALUE=C'***'
(IN)     /        SET-JV-LINK JV-NAME=JV.MON
(IN)     /        ENTER-JOB FROM-FILE=TEST.ENTER,MONJV=JV.MON,JOB-CLASS=
JCBATCH
(OUT)    %  JMS0066 JOB 'TESTJV' ACCEPTED ON 03-08-08 AT 09:22, TSN = 0FFJ
(IN)     /.SKIP1  SHOW-JV JV=JV.MON
(OUT)  $S 00FFJ4V05  J0672003-08-08072201
(  )
(IN)    /       WAIT-EVENT *JV(CONDITION=((JV.MON,1,2)=C'$R'),
TIME-LIMIT=90,TIMEOUT-LABEL=SKIP1)
(OUT)    %  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 09:22:01
(OUT)    %  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 09:22:01
(IN)     /.PRUEF  MODIFY-JV-CONDITIONALLY JV=(JV.DO.1),
IF-VALUE=C'GESTARTET',SET-VALUE=C'UNDERSTOOD',LABEL=WEIT1
(IN)     /        SKIP-COMMANDS TO-LABEL=PRUEF
(IN)     /.PRUEF  MODIFY-JV-CONDITIONALLY JV=(JV.DO.1),
IF-VALUE=C'GESTARTET',SET-VALUE=C'UNDERSTOOD',LABEL=WEIT1
(IN)     /        SKIP-COMMANDS TO-LABEL=PRUEF
(IN)     /.PRUEF  MODIFY-JV-CONDITIONALLY JV=(JV.DO.1),
IF-VALUE=C'GESTARTET',SET-VALUE=C'UNDERSTOOD',LABEL=WEIT1
(IN)     /.WEIT1  SHOW-JV JV=JV.DO.1
(OUT)    UNDERSTOOD
(IN)     /        SHOW-JV JV=JV.E.1
(OUT)    ***
(IN)     /        SHOW-JV JV=JV.MON
(OUT)  $R 00FFJ4V05  J0672003-08-08072201
(  )
(IN)     /        SHOW-JOB-STAT *MONJV(JV.MON)
(OUT)    TSN:     0FFJ      TYPE:    2 BATCH    NOW:     2003-08-08.092201
(NL)     JOBNAME: TESTJV     PRI:     9 255      SPOOLIN: 2003-08-08.0922
(NL)     USERID:  COGNITAS   JCLASS:  JCBATCH    LOGON:   2003-08-08.0922
(NL)     ACCNB:   89001      CPU-MAX:    200     CPU-USED:000000.0239
(NL)     REPEAT:  NO         RERUN:   NO         FLUSH:   NO
(NL)     MRSCAT:             HOLD:    NO         START:   SOON
(NL)     TID:     0001005F   UNP/Q#:    00/001
(NL)     CMD:     EXECUTE
(NL)     ORIGFILE::4V05:$COGNITAS.TEST.ENTER
(NL)     MONJV:   :4V05:$COGNITAS.JV.MON
(IN)     /        WAIT-EVENT *JV(CONDITION=((JV.DO.1)=C'CARRY ON'),
(  )              TIME-LIMIT=15,TIMEOUT-LABEL=WEIT1)
(OUT)    %  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 09:22:01
(OUT)    %  CJC0021 WAIT COMMAND: CONDITION = TRUE AT 09:22:01
```

```
(IN)     /.WEIT2  MODIFY-JV JV=JV.E.1,SET-VALUE=C'PROGRAM CONTINUE'
(IN)     /.SKIP3  SKIP-COMMANDS TO-LABEL=ENDE,
(  )              IF=*JV(CONDITION=((JV.E.1,12,3)=C'END'))
(OUT)    %  CJC0011 SKIP COMMAND: CONDITION = FALSE
(IN)     /        SHOW-USER-STATUS
(OUT)    NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT#
(NL)              OFFF 3 DIALOG   0 240      0.1275   32767 89001
(NL)     COG2     OFFG 3 DIALOG   0 240      0.2489   32767 89001
(NL)     TESTJV   OFFJ 2 BATCH    9 255      0.0394     200 89001
(OUT)    %  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
(OUT)    %  SPS0420 RSO WARNING : SOME RSO PRINT-JOBS CANNOT BE DISPLAYED
(IN)     /        SHOW-JV JV=JV.E.1
(OUT)    NORMALES   ENDE
(IN)     /        SHOW-JV JV=JV.MON
(OUT)    $R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV
*** PROGRAM CONTINUED ***
(  )
(IN)     /        WAIT-EVENT UNTIL=*JV(TIME-LIMIT=45,TIMEOUT-LABEL=SKIP3)
(OUT)    %  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 09:22:01
(OUT)    %  CJC0022 WAIT COMMAND: TIMEOUT AT 09:22:46, SKIP TO TIMEOUT LABEL
OR NEXT STEP
(IN)     /.SKIP3  SKIP-COMMANDS TO-LABEL=ENDE,
IF=*JV(CONDITION=((JV.E.1,12,3)=C'END'))
(OUT)    %  CJC0010 SKIP COMMAND: CONDITION = TRUE
(IN)     /.ENDE   REMARK **** TESTJOB ENDED ****
(IN)     /        SHOW-JV JV=JV.MON
(OUT)    $R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV
*** PROGRAM ENDED ***
(  )
(IN)     /        ASSIGN-SYSLST TO=*PRIMARY
```

Runtime listing OUT.E.TESTJV for job TEST.ENTER

```
/        SET-JV-LINK JV-NAME=JV.DO.1
/        SET-JV-LINK JV-NAME=JV.PROG
/        MODIFY-JV JV=JV.DO.1,SET-VALUE=C'GESTARTET'
/        START-EXE FROM-FILE=(LIB=ASS.PLAMLIB,ELEM=TESTJV),
TEST-OPTIONS=*AID,MONJV=JV.PROG
%  BLS0517 MODULE 'TESTJV' LOADED
*** TID: 0001005F *** TSN: 0FFJ
*****************************************************************************
*********************
SRC_REF:    78 SOURCE: TESTJV  PROC: TESTJV
*****************************************************************************
*********
LNKBER          = |.*ENTER
:4V05:$COGNITAS.JV.E.1...............................|
*** CSWJV EXECUTED ***
$R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV            ***
CSWJV EXECUTED ***
*** PROGRAM CONTINUED ***
$R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV            ***
PROGRAM CONTINUED ***
*** PROGRAM FINAL SPURT ***
$R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV            ***
PROGRAM FINAL SPURT ***
/        SHOW-JV JV=JV.MON
$R 00FFJ4V05    J0672003-08-080722012003-08-08072201TESTJV            ***
PROGRAM ENDED ***
/        SHOW-JV JV=JV.PROG
$T        P
/        SKIP-COMMANDS TO-LABEL=ENDE
/.ENDE   WAIT-EVENT UNTIL=*JV(TIME-LIMIT=120,TIMEOUT-LABEL=TIME)
%  CJC0020 WAIT COMMAND: TASK ENTERED WAIT STATE AT 09:22:01
%  CJC0022 WAIT COMMAND: TIMEOUT AT 09:24:01, SKIP TO TIMEOUT LABEL OR NEXT
STEP
/.TIME   MODIFY-JV JV=(JV.E.1,12,4),SET-VALUE=C'ENDE'
/        EXIT-JOB SYSTEM-OUTPUT=*NONE
%  EXC0419 /LOGOFF AT 0924 ON 03-08-08 FOR TSN '0FFJ'
%  EXC0421 CPU TIME USED: 0.0540
```

6 Messages

6.1 Command level

At the command level, error messages are preceded by a 7-character message code, comprising three characters indicating the message class and four characters for the error code. In the case of messages for job variable functions, the message class may be either JVS, JPM or CJC.
The table below shows the message codes and the components from which the respective messages originate.

Message class	Range of error codes	Error reported by/during
CJC	0000 - 0200	Conditional job control
JPM	0200 - 0500	Job/program monitoring
JVS	0400 - 040F	Catalog Management System
JVS	0410 - 041F	Catalog Management System
JVS	0420 - 042F	PAM
JVS	0430 - 043F	Catalog Management System
JVS	0440 - 044F	CREATE-JV/MODIFY-JV-ATTRIBUTES (CATJV) processing
JVS	0450 - 045F	SET-/REMOVE-/SHOW-JV-LINK (DCLJV) processing
JVS	0460 - 046F	DELETE-JV (ERAJV) processing
JVS	0470 - 047F	SHOW-JV/MODIFY-JV-CONDITIONALLY (GETJV) processing
JVS	0480 - 048F	MODIFY-JV/MODIFY-JV-CONDITIONALLY (SETJV) processing
JVS	0490 - 049F	SHOW-JV-ATTRIBUTES (STAJV) processing
JVS	04A0 - 04AF	General command processing
JVS	04B0 - 04BF	Various JV processing operations
JVS	04C8 - 04CA	Catalog Management System

Table 32: Message code and original component (part 1 of 2)

Message class	**Range of error codes**	**Error reported by/during**
JVS	04D0 - 04D8	MONJV handler
JVS	04E0 - 04E1	Command processor

Table 32: Message code and original component (part 2 of 2)

The command HELP-MSG-INFORMATION displays the message for the specified message code after SYSOUT in the current job language, from the message file activated for JV. With INFORMATION-LEVEL=*MAXIMUM, additional text and measurements are displayed. In the operand LANGUAGE a language code (D for German [Deutsch] or E for English) can be specified for the desired output language.

You can also find the JV messages on the manual server (URL: http://manuals.ts.fujitsu.com) by means of an HTML application and on the “BS2000/OSD SoftBooks” DVD.

6.2 Macro level

At macro level, a return code is always placed in the rightmost two bytes of register 15 or in a field provided for this purpose in the parameter list of the called macro (in the standard header). The meanings of the various return codes are commented on in the IDEJVS macro (leftmost byte, SI; rightmost byte, RS) or in the standard header of the called macro:

```
  MELD     IDEJVS
1 ***********************************************************************
1 *        VERSION 401
1          #INTF REFTYPE=REQUEST,                                       C
1                INTNAME=ERRJV,INTCOMP=000
1 ***********    JVS RETURN CODES    ******************************
1 IDRC0000 EQU   X'0000' REQUESTED JVS FUNCTION PROCESSED SUCESSFULLY.
1 *
1 *   JVS RETURN CODES FROM CMS
1 *
1 IDRC0401 EQU   X'0401' REQUESTED CATALOG NOT ACCESSIBLE.
1 IDRC0402 EQU   X'0402' REQUESTED CATALOG IN QUIET MODE.
1 IDRC0403 EQU   X'0403' MRSCAT CONTAINS ILLEGAL INFORMATION.
1 IDRC0404 EQU   X'0404' SYSTEM ERROR IN CMS.                       202
1 IDRC0405 EQU   X'0405' SYSTEM ERROR IN MC.
1 IDRC0406 EQU   X'0406' MASTER EXCH OPERATION ABORT.               103
1 IDRC0407 EQU   X'0407' MASTER EXCH WRITE DENIED.                  103
1 IDRC0410 EQU   X'0410' RC FROM SUBFUNCTION.                       103
1 IDRC041F EQU   X'041F' TASK RP01 NO DUMP.                         201
1 IDRC0412 EQU   X'0412' REQUESTED CATALOG NOT FOUND.               202
1 IDRC0430 EQU   X'0430' SYSTEM ERROR IN CMS ($GETMEM).
1 IDRC0431 EQU   X'0431' INVALID PARAMETER.
1 IDRC0432 EQU   X'0432' SYSTEM ERROR IN CMS.
1 IDRC0433 EQU   X'0433' REQUESTED JOB VARIABLE NOT CATALOGED.
1 IDRC0434 EQU   X'0434' SYSTEM ERROR IN CMS.
1 IDRC0435 EQU   X'0435' JOB VARIABLE NOT SHARABLE.
1 IDRC0436 EQU   X'0436' SYSTEM ERROR IN CMS.
1 *   IDRC0437 EQU   X'0437' SYSTEM ERROR IN CMS.
1 IDRC0438 EQU   X'0438' 6 OR LESS BLOCKS REMAINING FOR CATALOG.
1 IDRC0439 EQU   X'0439' NO MORE SPACE AVAILABLE FOR CATALOG.
1 IDRC043B EQU   X'043B' SYSTEM ERROR IN CMS.
1 IDRC043C EQU   X'043C' CATALOG FILE SPACE EXHAUSTED.
1 IDRC043D EQU   X'043D' SYSTEM ERROR: TSOSCAT DESTROYED.
1 IDRC043E EQU   X'043E' JV TO BE CREATED ALREADY EXISTS.
1 *
1 *   JVS RETURN CODES FROM DQPAM
1 *
1 *   IDRC0421 EQU   X'0421' SYSTEM ERROR: NO I/O SLOT AVAILABLE.    005
1 IDRC0422 EQU   X'0422' SYSTEM ERROR IN CMS.
1 IDRC0423 EQU   X'0423' SYSTEM ERROR IN PAM.
```

```
1 IDRC0424 EQU   X'0424' SYSTEM ERROR: SYSTEM ADDRESS SPACE EXHAUSTED.
1 IDRC0425 EQU   X'0425' SYSTEM ERROR IN CMS.
1 IDRC0426 EQU   X'0426' SYSTEM ERROR IN CMS.
1 IDRC0427 EQU   X'0427' SYSTEM ERROR IN PAM.
1 IDRC0428 EQU   X'0428' SYSTEM ERROR IN PAM.                        005
1 IDRC0429 EQU   X'0429' SYSTEM ERROR IN PAM.
1 *   IDRC042B EQU   X'042B' SYSTEM ERROR IN PAM.                    005
1 IDRC042C EQU   X'042C' SYSTEM ERROR IN PAM.
1 IDRC042D EQU   X'042D' SYSTEM ERROR IN PAM.                        005
1 IDRC042F EQU   X'042F' SYSTEM ERROR: RESIDENT PAGE SPACE EXHAUSTED.
1 *
1 *   JVS RETURN CODES FROM JVCATEX
1 *
1 IDRC0440 EQU   X'0440' INVALID JVNAME1.
1 IDRC0441 EQU   X'0441' INVALID JVNAME2.
1 IDRC0442 EQU   X'0442' NEGATIVE RETENTION PERIOD.
1 *   IDRC0443 EQU   X'0443' STATE=NEW AND ACCESS=READ.              100
1 IDRC0444 EQU   X'0444' JVNAME1 OR JVNAME2 ALREADY CATALOGED.
1 IDRC0445 EQU   X'0445' INVALID RETENTION PERIOD SPECIFICATION.
1 *   IDRC0446 EQU   X'0446' RENAME JV NOT SUCCESSFUL.               005
1 IDRC0447 EQU   X'0447' JVNAME1 IS IN USE BY CJC.                   001
1 IDRC0448 EQU   X'0448' MONJV MAY NOT BE CHANGED.                   001
1 IDRC0449 EQU   X'0449' ONLY STANDARD ATTRIBUTES FOR TEMPJV ALLOWED 800
1 IDRC044A EQU   X'044A' GUARD NAME INVALID                          200
1 IDRC044B EQU   X'044B' MANAGEMENT-CLASS INVALID                    200
1 IDRC044C EQU   X'044C' ERROR IN DEFAULT PROTECTION                 300
1 *
1 *   JVS RETURN CODES FROM JVDCLEX, JVRELEX, JVLNKEX AND JVCSWEX
1 *
1 IDRC0450 EQU   X'0450' INVALID LINKNAME.
1 IDRC0451 EQU   X'0451' NO TFT EXISTING.                            800
1 *   IDRC0452 EQU   X'0452' INVALID JOBVAR NAME.                    003
1 IDRC0453 EQU   X'0453' NO USER AREA.                               800
1 *   IDRC0454 EQU   X'0454' INVALID PARAMETER.                      003
1 IDRC0455 EQU   X'0455' AREA SIZE TO SMALL                          800
1 IDRC0456 EQU   X'0456' COMPARISON IS FALSE                         801
1 IDRC0457 EQU   X'0457' REMOVE ALL ENTRIES ?                        103
1 IDRC0458 EQU   X'0458' REQU WITHDRAWN BY USER.                     103
1 *
1 *   JVS RETURN CODES FROM JVERAEX
1 *
1 IDRC0460 EQU   X'0460' INVALID ERASE REQUEST.
1 IDRC0461 EQU   X'0461' ERASE OF SOME JOB VARIABLES IN ERROR.
1 IDRC0462 EQU   X'0462' SEVER PROCESSING BY P1 MACRO CALLER REQUESTED.
1 IDRC0463 EQU   X'0463' ERASE ERROR ON JV.
1 IDRC0464 EQU   X'0464' ERAJV ERROR. USER HAS STILL JOB VARIABLE
1 *                      CATALOG ENTRIES.
1 IDRC0465 EQU   X'0465' ERASE ALL JV'S ON USERID ?                  950
```

```
1 IDRC0466 EQU   X'0466' MONJV IS PROTECTED                           800
1 IDRC0468 EQU   X'0468' ERASE ALL JV'S ON PUBSET ?                   950
1 IDRC0469 EQU   X'0469' ERASE A SINGLE JV ?                          950
1 IDRC046A EQU   X'046A' ERASE REQUEST WITHDRAWN BY CALLER            950
1 *
1 *   JVS RETURN CODES FROM JVGETEX
1 *
1 IDRC0470 EQU   X'0470' JV VALUE CONTAINS FEWER BYTES THAN REQUESTED.
1 IDRC0471 EQU   X'0471' JOB VARIABLE $SYSJV.LASTMSG NOT ACCESSIBLE. 401
1 IDRC0472 EQU   X'0472' INVALID SPECIAL JOB VARIABLE.
1 IDRC0474 EQU   X'0474' JV VALUE TRUNCATED.
1 IDRC0475 EQU   X'0475' THE SIZE SPECIFIED FOR THE AREA IN THE GETJV
1 *                      MACRO IS LESS THAN 4 BYTES.
1 *
1 *   JVS RETURN CODES FROM JVSETEX
1 *
1 IDRC0480 EQU   X'0480' SPECIAL JOB VARIABLE MAY NOT BE SET.
1 IDRC0481 EQU   X'0481' A NON-PRIVILEGED CALLER HAS REQUESTED A
1 *                      PRIVILEGED SETJV FUNCTION.
1 IDRC0482 EQU   X'0482' THE ADDRESS OF THE JV VALUE IS MISSING.
1 IDRC0483 EQU   X'0483' THE LENGTH OF THE JV VALUE EXCEEDS 256 BYTES.
1 IDRC0484 EQU   X'0484' MORE THAN 1 PRIVILEGED REQUEST HAS BEEN
1 *                      SPECIFIED IN A SINGLE CALL.
1 IDRC0485 EQU   X'0485' JVNAME2 AND VALUE ADDRESS ARE BOTH SPECIFIED.
1 *   IDRC0486 EQU   X'0486' JVNAME1 AND JVNAME2 ARE NOT OF SAME TYPE120
1 IDRC0487 EQU   X'0487' FIRST BYTES OF MONJV ARE PROTECTED.          800
1 *
1 *   JVS RETURN CODES FROM JVSTAEX
1 *
1 IDRC0490 EQU   X'0490' THE AREA SIZE PARAMETER IS TOO SMALL TO CONTAIN
1 *                      THE CATALOG ENTRY.
1 IDRC0491 EQU   X'0491' INVALID OPERAND IN SELECTION LIST
1 *
1 *   JVS RETURN CODES FROM JVXXXSY, JVSV133, JVSV190, JVSYSCM, JVSYSP2
1 *
1 IDRC04A0 EQU   X'04A0' FUNCTIONAL UNIT "JOB VARIABLE SERVICES" (JVS)
1 *                      NOT SELECTED.
1 IDRC04A1 EQU   X'04A1' SYNTAX ERROR IN COMMAND PARAMETER.
1 IDRC04A2 EQU   X'04A2' JV HAS BEEN ERASED.
1 IDRC04A3 EQU   X'04A3' ERASE ERROR ON JV.
1 IDRC04A4 EQU   X'04A4' INVALID FUNCTION CODE SPECIFIED.
1 IDRC04A5 EQU   X'04A5' A NON-PRIVILEGED CALLER HAS REQUESTED A
1 *                      PRIVILEGED ENCRYPTION OPTION.
1 IDRC04A6 EQU   X'04A6' SYSTEM ERROR IN WROUT
1 IDRC04A7 EQU   X'04A7' INVALID ADDRESS IN REGISTER 1 DURING SVC133.
1 *   IDRC04A8 EQU   X'04A8' INVALID INTERACTIVE MODE.                007
1 IDRC04A9 EQU   X'04A9' LABEL NOT FOUND.                             801
1 IDRC04AA EQU   X'04AA' INVALID PL IN CURRENT MODE (GET,SET,STA)    902
```

```
1 IDRC04AB EQU   X'04AB' INVALID PARAMETER IN MACRO PARAMETER LIST   950
1 *              (INVALID FLAGS OR OLD INTERFACE USED NEW FUNTIONS
1 IDRC04AC EQU   X'04AC' INVALID CONVERSION OF TIME UTC TO LT        101
1 IDRC04AD EQU   X'04AD' ERROR DURING VARIABLE PRODUCTION            120
1 *
1 *   JVS RETURN CODES FROM DIFFERENT JVS MODULES
1 *
1 IDRC04B0 EQU   X'04B0' SIZE FIELD ZERO OR AREA ADDRESS NOT SPECIFIED.
1 IDRC04B1 EQU   X'04B1' PASSWORD HAS NOT BEEN PROVIDED.
1 IDRC04B2 EQU   X'04B2' REQUESTED JOB VARIABLE OR REQUESTED SUBSTRING
1 *                      IS EMPTY.
1 IDRC04B3 EQU   X'04B3' INCORRECT SYNTAX-JVNAME.
1 IDRC04B4 EQU   X'04B4' LINKNAME NOT PREVIOUSLY DEFINED.
1 IDRC04B5 EQU   X'04B5' SYSTEM ERROR: $GETMEM ERROR.                999
1 IDRC04B6 EQU   X'04B6' EXPIRATION DATE ERROR.
1 IDRC04B7 EQU   X'04B7' SYSTEM ERROR: $RETMEM ERROR.                999
1 IDRC04B8 EQU   X'04B8' ONLY READ ACCESS IS ALLOWED.
1 IDRC04B9 EQU   X'04B9' ILLEGAL SUBSTRING IN GET- OR SETJV.
1 *   IDRC04BA EQU   X'04BA' ILLEGAL SYNTAX FOR NON-NUMERIC JV.      120
1 IDRC04BB EQU   X'04BB' ILLEGAL SYNTAX FOR NUMERIC JV.              750
1 IDRC04BC EQU   X'04BC' NOT ALL JVS ARE ERASED OR DISPLAYED         950
1 IDRC04BD EQU   X'04BD' ERROR OCCURED WHILE USING ACCESS-FUNCTION   001
1 *                      TO SYSTEM-TABLE (E.G. TCB)                  001
1 IDRC04BE EQU   X'04BE' USERID DOES NOT EXIST                       002
1 IDRC04BF EQU   X'04BF' JV PROTECTED BY ACL                         004
1 *
1 *   JVS RETURN CODES FROM CMS
1 *
1 IDRC04C8 EQU   X'04C8' JV LIMIT EXCEEDED                           100
1 IDRC04CA EQU   X'04CA' JV LIMIT ERROR                              102
1 *
1 *   JVS RETURN CODES FROM JVSMJVH                                  800
1 *
1 IDRC04D0 EQU   X'04D0' JV NOT ACCESSIBLE.                          800
1 IDRC04D1 EQU   X'04D1' JV NOT ASSIGNED.                            800
1 IDRC04D2 EQU   X'04D2' JV ALREADY ASSIGNED.                        800
1 IDRC04D3 EQU   X'04D3' JV CANNOT BE CREATED.                       800
1 IDRC04D4 EQU   X'04D4' JV INCORRECTLY SPECIFIED.                   800
1 IDRC04D5 EQU   X'04D5' JV CATALOG ENTRY IS LOCKED.                 800
1 IDRC04D6 EQU   X'04D6' JV NOT SUPPLIED WITH TSN.                   800
1 IDRC04D7 EQU   X'04D7' GCF ERROR OCCURRED.                         400
1 IDRC04D8 EQU   X'04D8' NO LINKNAME FOR MONJV.                      801
1 IDRC04D9 EQU   X'04D9' GCF ERROR TO CONSOLE.                       400
1 *
1 *   JVS RETURN CODES TO CMD-PROCESSOR                              101
1 *
1 IDRC04E0 EQU   X'04E0' CORRECT AND RETRY.                          101
1 IDRC04E1 EQU   X'04E1' WAIT AND RETRY.                             101
```

```
1 *
1 *   JVS RETURN CODES FROM CMS
1 *
1 IDRC14A4 EQU   X'14A4' CATALOG INDEX MAX SIZE.                    400
1 IDRC14A5 EQU   X'14A5' CATALOG INDEX DESTROYED.                   400
1 *********************************************************************
1          SPACE
```

The return codes listed correspond in most cases to the error codes of system messages and can be supplemented to form a message code by adding the message class. You can thus obtain detailed information in the manner described in section “Command level” on page 227.

No corresponding system message exists for the following return codes. They have the following meanings:

X'0000'	Execution without error.
X'0450'	Invalid link name.
X'0453'	User area could not be supplied.
X'0455'	User area is too small.
x'0456'	Job variable value is not equal to comparison value.
X'0461'	Error while deleting job variables. The jvid specified in the ERAJV matches (e.g. due to a wildcard specification) more than one object (catalog, user ID and/or JV). Not all of these objects (possibly none at all) could be processed correctly.
X'0462'	User requested REMOVE-USER processing (the IDJESEVR bit was set in the ERAJV parameter list).
X'0475'	The area length is less than 4 bytes.
X'0481'	Non-privileged user requested privileged SETJV function.
X'0482'	Address for job variable value is missing.
X'0483'	The length in the record length field of the job variable value is greater than 260 bytes or less than 4 bytes.
X'0484'	More than one privileged request was submitted in one call.
X'0485'	Both “jvname2” and a value address were specified.
X'0490'	Length of defined area is insufficient for catalog entry.
X'0491'	Invalid selection operand was specified (JVSEL).
X'04A4'	An invalid function code was specified (SVC 133).

X'04A5'	Non-privileged user requested privileged coding function.
X'04A7'	Invalid address in register 1 with SVC 133 or SVC 190.
X'04AA'	User is in 31-bit mode but used a 24-bit parameter list (GETJV, SETJV or STAJV).
X'04AB'	A new parameter was used in an old parameter list or the user set the flag byte to an invalid value (ERAJV, STAJV, etc.).
X'04B0'	Field length is zero or area address is not defined.
X'04B9'	An invalid value range was specified.

The following return codes are returned in the event of incorrect initialization of the standard header (PARMOD=31):

X'0001FFFF':	The unit and/or function number in the header is not supported.
X'0003FFFF':	The version number in the header refers to an interface version which is not supported.
X'0004FFFF':	The parameter list is not aligned on a word boundary.

7 The privileged user

Privileged users are users authorized to perform systems support, usually working under the user ID TSOS. When the software product SECOS is installed, the systems support functions are distributed over different user IDs. Unless expressly stated otherwise, “privileged user” as employed in the following always refers to systems support under the TSOS user ID.

7.1 Installation

The software product Job Variables (JV) V15.0A is used in the currently released versions of BS2000/OSD-BC. It is a chargeable software product not included in the basic configuration of BS2000.
Job Variables 15.0A is supplied as a dynamically loadable subsystem (as defined by DSSM).

The following components are supplied with JV 15.0A:

File name	**Contents**
SYSFGM.JV.150.D	Release Notice (German)
SYSFGM.JV.150.E	Release Notice (English)
SYSRME.JV.150.D	Readme file (German)
SYSRME.JV.150.E	Readme file (English)
SYSLNK.JV.150	Link and load module (LLM) for /390 systems
SPMLNK.JV.150	Link and load module (LLM) for SX servers
SKMLNK.JV.150	Link and load module (LLM) for SQ servers
SYSLIB.JV.150	User macro library
SIPLIB.JV.150	TPR macro library
SYSMES.JV.150	Message file (see chapter “Messages” on page 227)
SYSRMS.JV.150	Delivery package for RMS
SYSSDF.JV.150	Syntax file (see “Syntax files” on page 236)
SYSSSC.JV.150	Subsystem declaration (see “Subsystem declarations” on page 236)

The installation must be done using IMON.

Subsystem declarations

The JV subsystem is loaded by DSSM and is available at “system ready” time. The JV subsystem must already be declared in the subsystem catalog to permit this. The requisite subsystem declarations are contained in the file SYSSSC.JV.150. Subsystem catalog generation is described in the manual “Subsystem Management” [7].

At initialization the JV subsystem opens the system file $TSOS.SYSCAT.JV on the home pubset for internal use. If the file does not exist, it is created. The file is only closed when the JV subsystem is closed.

Syntax files

The JV product is supplied with the syntax file SYSSDF.JV.150, which is entered as a subsystem syntax file in the SDF parameter file by IMON.

The following applies to syntax files created by a user group or individual user whenever there is a change of version:

– If these files contain JV commands which have been modified with SDF-A, they must be created again as new files.
– The same applies to group syntax files which are used with HIERARCHY=NO and contain JV commands.

The management of system and group syntax files and the SDF-I utility routine are described in the manual “SDF Dialog Interface” [2].

7.2 Privileged access rights

The privileged user is treated in the same way as the owner in respect of the job variables of other users (it can be limited, see “Restricted TSOS co-ownership” on page 24). This also applies to the operator, who is authorized to issue the SHOW-JV, MODIFY-JV and SHOW-CJC-STATUS commands at the console. With basic ACL protection, the access rights defined for OWNER are applicable.
The privileged user can use wildcards in all commands and macros which allow wildcards in the JV name, as well as within the user ID (e.g. SHOW-JV-ATTRIBUTES or STAJV macro).

During output of the JV entry (SHOW-JV-ATTRIBUTES or STAJV macro), specified passwords are included in the output for the privileged user. The privileged user can modify a JV entry (MODIFY-JV-ATTRIBUTES or CATJV macro) without specifying passwords. When deleting job variables (DELETE-JV or ERAJV macro), the privileged user is able to ignore the password protection (IGNORE-PROTECTION operand).

The privileged user is also authorized to create job variables under any user ID (CREATE-JV or CATJV macro), and can also access all temporary job variables in the system.
The systems support group can transfer system administrator functions to the operator (see “Introductory Guide to Systems Support” [3]). This means that the operator may also issue other commands in addition to SHOW-JV and MODIFY-JV (such as CREATE-JV and DELETE-JV) at the console, with the same privileges.

7.3 System file SYSCAT.JV

At initialization, the JV subsystem opens the system file $TSOS.SYSCAT.JV on the home pubset for internal use. If the file does not exist, it is created. The file is only closed when the JV subsystem is closed.

7.4 Monitoring in pubset management

When managing pubsets, systems support can have the following functions monitored by job variables:

- “Import pubset”
- “Export pubset”
- “Force pubset export”

When a shared pubset is imported, a job variable is created which indicates the availability of this pubset.

The following status values are set in a monitoring job variable during import and export of pubsets:

Value	Function	Meaning/reason for assigning the value
$A	Export	Export aborted due to error or CANCEL-PUBSET-EXPORT command
	Import	Import aborted due to error
$E	Export	Export job successfully initiated
$I	Import	Import job successfully initiated
$R	Import	Pubset successfully imported
$T	Export	Pubset successfully exported
$W	Import	During import of a shared pubset the processor is waiting for the acknowledgment from the master

Table 33: Status values when importing/exporting a monitoring job variable

In shared pubset operation within a multiprocessor network, a shared pubset-specific job variable is created on each home pubset of a sharer in order to monitor operations in the network. .

Value	Meaning/reason for assigning the value
$R	Shared pubset available. Master switchover successfully completed.
$T	Shared pubset not available, e.g. due to failure of master.
$A	Master switchover aborted due to error, e.g. failure of the master and unsuccessful declaration of a new master from the various slave processors.
$C	Due to failure or shutdown of the master, a master switchover is performed on the public volume.

Table 34: Monitoring job variables in shared-pubset mode

An overview of the commands in which systems support can monitor pubset management by means of job variables is presented in table 35 below.
For more information on pubset management, refer to the manuals “Introductory Guide to Systems Support” [3] and “HIPLEX MSCF” [8].

Notes on using a monitoring job variable

– The job variable must not be write-protected.

– If the job variable is **not** cataloged, message DMS0383 is output and the import/export job is executed without a monitoring job variable.

– If the job variable is protected by a password, it can be declared as a monitoring job variable only if the user submitting the job has already entered the password in the password table (with ADD-PASSWORD) or specifies it in the JV-PASSWORD operand of the following commands.

Command	Operands relevant to monitoring of pubset management
EXPORT-PUBSET exports a previously imported pubset	**MONJV=*NONE/<filename 1..54 without-gen>** defines the monitoring JV. **JV-PASSWORD=NONE/<c-string 1..4>/<x-string 1..8>/<integer -2147483648..21484836479>** defines a password for the monitoring JV.
Note on EXPORT-PUBSET: The command generates a job that performs the pubset export. The pubset to be exported is set to “inaccessible”. The job waits until all jobs reserving resources have released their allocation (open files, or reservation of files with SECURE-RESOURCE-ALLOCATION). The export job returns information about the number of jobs still reserving the pubset. The task sequence numbers (TSNs) can be queried using the SHOW-PUBSET-PARAMETER command. Following termination of the wait state the user catalog is closed and all resources are released. Messages generated by the job are output at the operator terminal. The wait state during export processing can be terminated with the operand TERMINATE-JOBS=YES. This causes all jobs still occupying resources to be terminated in an orderly manner. If the wait state cannot be exited, this can be enforced by means of the FORCE-CATALOG-EXPORT command. The change in availability is reported to all active processors of the multiprocessor network. The export job can be monitored by means of a job variable (see **MONJV**). Conditional job control (CJC) can be used to respond to the various processing states.	
FORCE-PUBSET-EXPORT forces export of a pubset.	**MONJV=*NONE/<filename 1..54 without-gen>** defines the monitoring JV. **JV-PASSWORD=NONE/<c-string 1..4>/<x-string 1..8>/<integer -2147483648..21484836479>** defines a password for the monitoring JV.

Table 35: Commands in which job variables for monitoring in pubset management can be defined (part 1 of 2)

Command	Operands relevant to monitoring of pubset management
IMPORT-PUBSET imports a pubset.	**MONJV=*NONE/<filename 1..54 without-gen>** defines the monitoring JV. **JV-PASSWORD=NONE/<c-string 1..4>/<x-string 1..8>/<integer -2147483648..21484836479>** defines a password for the monitoring JV.
Note on IMPORT-PUBSET: The import job can be monitored using a job variable (see **MONJV)**. Conditional job control (CJC) can be used to respond to the various processing states. If a shareable pubset was imported, its availability is additionally logged in a separate job variable. During the import process this is reset or created as a new job variable on the home pubset of the importing processor: The default name is :**<cat-id home>:$TSOS.SYS.PVS.<cat-id shared-pubset>.MASTER.CONTROL** Possible status values: $R : Shared pubset available $T : Shared pubset not available (e.g. due to master processor crash)	

Table 35: Commands in which job variables for monitoring in pubset management can be defined (part 2 of 2)

7.5 Job variables as the object of system monitoring

When the software product SECOS is used, job variables belong to the objects which can be monitored with the function unit SAT (JOB VARIABLES object).

SAT allows logging of accesses to the Job Variables object. Logging of a particular access (e.g. reading a JV) occurs when the security administrator has allowed the Job Variables object to be used for monitoring. In addition, the security administrator can make logging of a particular access dependent on the result: successful (SUCC) or unsuccessful (FAIL). The result (SUCC or FAIL) and the fully qualified or partially qualified job variable name or a wildcard string are logged for job variables. Logging of the return information from JV can also be allowed. The default value here is *NONE; this remains unchanged in the case of the result "successful access" (SUCC).
General errors occurring during access are not logged (syntax error, parameter list error, incorrect job variable name). Accesses by the operator are similarly not logged by SAT (except to the CONSLOG file).
System monitoring with SAT is described in detail in the manual "SECOS" [9].

The events described below can be selected for the JOB VARIABLES object. The short name for the event is given first, followed by the commands and macros which can trigger the event:

JVC	Create job variable entry (and the protection attributes): CREATE-JV command or CATJV macro with STATE=NEW; also SET-JV-LINK command or DCLJV macro if a non-existent job variable is set up.
JVM	Modify protection attributes of a job variable: MODIFY-JV-ATTRIBUTES command or CATJV macro with STATE=UPDATE.
JVR	Rename job variable: NEW-NAME is specified in the MODIFY-JV-ATTRIBUTES command or a second JV name is given in the CATJV macro with STATE=UPDATE. If protection attributes are also modified at the same time, then a second SAT record is written for the event JVM.
JVA	Rename job variable with reconstruction via the ARCHIVE utility routine.
JVD	Delete job variable entry (and the protection attributes): DELETE-JV command and ERAJV macro. When using a partially qualified JV name or wildcards, the list of the affected job variables is not logged. During deletion, a SAT record is written for an affected job variable. If only the value of the job variable is deleted (OPTION=DATA or DATA=YES), this is only a write access (see event JVS).

JVQ	List information about job variables (and the protection attributes): SHOW-JV-ATTRIBUTES command or STAJV macro. A specified partially qualified JV name or wildcard string is also logged. A SAT record is written for each affected job variable.
JVG	Read job variable value: SHOW-JV command or GETJV macro, when used in conditional expressions and in the job variable substitution.
JVS	Write job variable value: MODIFY-JV, MODIFY-JV-CONDITIONALLY, MODIFY-MONJV commands or SETJV, CSWJV, TIMJV macro calls. If the value to be set is taken over from a job variable, then a further SAT record is written for read access to this job variable (see event JVG).

Logging of a specific access to the JOB VARIABLES object can be made dependent on the following information:

JVNAME	Fully or partially qualified job variable name
JVPATRN	Wildcard pattern
NEWJV	New job variable name
JVSRC	Return code information

The table below shows which information is mandatory (M), optional (O) or not essential ("-") in order to enable certain events to be logged for the JOB VARIABLES object

Information	**Event**							
	JVA	**JVC**	**JVD**	**JVG**	**JVM**	**JVQ**	**JVR**	**JVS**
JVNAME	M	M	M	M	M	O [1]	M	M
JVPATRN	-	-	-	-	-	O [1)]	-	-
JVSRC	O	O	O	O	O	O	O	O
NEWJV	M	-	-	-	-	-	M	-

Table 36: Information dependencies for logging an access to the JOB VARIABLES object

[1] Either of the two is required.

Abbreviations

ACL	Access Control List
BACL	Basic Access Control List
CJC	Conditional Job Control
CMS	Catalog Management System
DSECT	Dummy SECTion
DSSM	Dynamic SubSystem Management
GUARDCOO	GUARD Co-Owner Protection
GUARDS	**G**enerally **U**sable **A**ccess Cont**R**ol A**D**ministration **S**ystem (access control monitor)
ISP	**I**nteractive String Processor (command decoder in BS2000 used previous to SDF)
JMS	Job Management System
JPM	Job Program Management
JV	Job Variable
JVS	Job Variable System
MONJV	MONitoring Job Variable
MRSCAT	Catalog of the pubsets imported on the processor; in a multiprocessor network, the participating processors are also entered in the catalog under their catalog IDs.
MSCF	Multi System Control Facility

MF	Macro Format
RS	Return Switch
SAT	Security Audit Trail
SDF	System Dialog Facility (standard command decoder)
SECOS	SEcurity COntrol System
SI	Secondary Indicator
TPR	Task PRivileged
TSOSCAT	System catalog
TSN	Task Sequence Number
TU	Task Unprivileged
UTC	Universal Time Coordinate (corresponds to Greenwich time)
VSN	Volume Serial Number

Related publications

The manuals are available as online manuals, see *http://manuals.ts.fujitsu.com*, or in printed form which must be paid and ordered separately at *http://manualshop.ts.fujitsu.com*.

[1] **BS2000/OSD-BC**
Commands
User Guide

[2] **SDF** (BS2000/OSD)
SDF Dialog Interface
User Guide

[3] **BS2000/OSD-BC**
Introductory Guide to Systems Support
User Guide

[4] **BS2000/OSD-BC**
Executive Macros
User Guide

[5] **BS2000/OSD-BC**
DMS Macros
User Guide

[6] **SDF-P** (BS2000/OSD)
Programming in the Command Language
User Guide

[7] **DSSM / SSCM**
Subsystem Management in BS2000/OSD
User Guide

[8] **HIPLEX MSCF** (BS2000/OSD)
BS2000 Processor Networks
User Guide

[9] **SECOS** (BS2000/OSD)

Security Control System - Audit
User Guide

[10] **SECOS** (BS2000/OSD)
Security Control System - Access Control
User Guide

[11] **ARCHIVE** (BS2000/OSD)
User Guide

[12] **HSMS / HSMS-SV** (BS2000/OSD)
Hierarchical Storage Management System
User Guide

[13] **SPOOL** (BS2000/OSD)
User Guide

[14] **BLSSERV**
Dynamic Binder Loader / Starter in BS2000/OSD
User Guide

[15] **openFT** (BS2000/OSD)
Enterprise File Transfer in the Open World
User Guide

[16] **SPACEOPT** (BS2000/OSD)
Disk Optimization and Reorganization
User Guide

[17] **VM2000** (BS2000/OSD)
Virtual Machine System
User Guide

Index