SDF-P V2.4A

Programming in the Command Language

Edition May 2007

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

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

Feel free to send us your comments by e-mail to manuals@fujitsu-siemens.com.

Certified documentation according DIN EN ISO 9001:2000

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 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

Preface

This Ready Reference provides you with a quick means of finding the information you need for generating S procedures. Detailed information is contained in the SDF-P User Guide.

The Ready Reference is divided into five parts:

- Overview and summary descriptions of the SDF-P functions
- Overview, general description of the return codes and summary descriptions of the SDF-P commands
- Description of the Assembler macro interface
- Description of the syntax for variable names
- Description of the metasyntax, data types and operators

Predefined functions

Overview

Function	**Brief description**
ACCOUNT( )	Returns the account number of the task
ARRAY-INDEX( )	Returns the value of an array index
BOOLEAN( )	Converts an expression to a Boolean value
CHARACTER-TO-INTEGER( )	Converts a character in EBCDI code to an integer
CHECK-DATA-TYPE( )	Checks the SDF data type in strings or operand values
COUNTER( )	Counts the number of times COUNTER() is called
CURRENT-TYPE( )	Returns the current data type of a simple variable
DATE( )	Returns the current date
DATE-VALUE( )	Outputs a specific date
DAY( )	Returns the name of the current day
ELAPSED-DAYS( )	Returns the number of days that exist between two given dates
EXPLICIT-CALL( )	Indicates the type of the procedure call
EXTEND-SDF-LIST( )	Adds an element to an SDF list
EXTRACT-FIELD( )	Removes a field from an input string
FILL( )	Expands a string using fill characters
FIRST-VARIABLE-NAME( )	Analyzes the structure of complex variables
FROM-C-LITERAL( )	Converts a C literal to the corresponding string value
FROM-X-LITERAL( )	Converts an X literal to its hexadecimal value
HASH-STRING( )	Encrypts an expression as a string
HASH-VALUE( )	Encrypts an expression as an integer value
HOME-CAT-ID( )	Returns the catalog ID of the home pubset
HOST( )	Returns the internal name of the system on which the procedure is running
INDEX( )	Returns the position of a search string within a full string
INSTALLATION-PATH( )	Outputs an installation path name of a file
INTEGER( )	Converts any expression to the data type INTEGER
INTEGER-TO-CHARACTER( )	Converts an integer to a character (C string)
INTEGER-TO-X-LITERAL( )	Converts an integer to a 4-byte long X literal which contains the coding of the integer (inverse function to X-LITERAL-TO-INTEGER)

Function	**Brief description**
IS-C-LITERAL( )	Checks whether the specified string is a C literal and can be converted to a string
IS-CATALOGED-FILE( )	Checks whether there is a catalog entry with the specified file name
IS-CATALOGED-JV( )	Checks whether there is a catalog entry with the specified job variable name
IS-DECLARED( )	Checks whether the specified simple or complex variable has already been declared
IS-EMPTY-FILE( )	Checks whether a file is empty when the value 0 is returned by HIGHEST-USED-PAGES( )
IS-INITIALIZED( )	Checks whether the specified variable is initialized
IS-INTEGER( )	Checks whether the specified string represents an integer value
IS-LIBRARY( )	Checks whether the specified file is cataloged as a PLAM library
IS-LIBRARY-ELEMENT( )	Checks whether or not the specified library element exists
IS-SDF-LIST( )	Analyzes whether a string is an SDF list and/or meets a criterion for an SDF list
IS-SDF-P( )	Checks whether SDF-P is loaded in the system
IS-SDF-STRUCTURE( )	Checks whether the specified string is an SDF structure
IS-VARIABLE-NAME( )	Checks whether the specified string is a syntactically correct variable name
IS-X-LITERAL( )	Checks whether the specified expression contains an X literal and can be converted using FROM-X-LITERAL( )
JOB-CLASS( )	Returns the job class of the current task
JOB-MONJV( )	Returns the name of the variable monitoring the job
JOB-NAME( )	Returns the job name of the current task
JV( )	Returns the contents of the specified job variable
LAYOUT-SCOPE( )	Returns the scope of the specified structure layout
LENGTH( )	Returns the length of the specified string
LIMIT( )	Indicates the maximum number of elements that the specified list variable can contain
LOGGING-MODE( )	Indicates whether the logging of commands or data has been enabled for the procedure call
LOWER-CASE( )	Converts all uppercase letters in the specified string to lowercase
MAINCODE( )	Returns the 7-byte error code contained in the return code
MONTH( )	Returns the name of the current month in the language specified
MSG( )	Returns the message text assigned to the specified message code

Function	Brief description
NEXT-VARIABLE-NAME( )	Analyzes the structure of complex variables
PROC-LEVEL( )	Returns the current nesting depth of the S procedure
PROCESSOR( )	Returns the processor name of the TIAM station
PROG-MONJV( )	Returns the name of the job variable monitoring the program
PROG-NAME( )	Returns the internal name (reduced to eight characters) of the program currently loaded
RENAME( )	Returns a new name
REPLACE( )	Replaces a substring with another string
RUN-PRIORITY( )	Returns the priority level of the current job
SDF-P-VERSION( )	Returns the name of the current version of SDF-P or SDF-P-BASYS
SDF-STRUCTURE-VALUE( )	Returns part or all of the contents of an SDF structure
SEARCH-LIST-INDEX( )	Searches for a string in a list
SESSION-NUMBER( )	Returns the session number of the system currently running
SIZE( )	Returns the number of elements contained in the complex variable specified
STATION( )	Returns the name of the TIAM station from which the procedure was started
STATION-TYPE( )	Returns the device type generated for the TIAM station from which the procedure was called
STD-CAT-ID( )	Returns the ID of the pubset which was assigned to the current user ID as the default pubset
STMT-SPINOFF( )	Indicates whether statement spin-off has been enabled for the program currently loaded
STRING( )	Converts an expression of the type INTEGER, BOOLEAN or STRING to the type STRING
SUBCODE1( )	Returns subcode1 (error class) of the current command return code
SUBCODE2( )	Returns subcode2 (additional information) of the current command return code
SUBLIST( )	Returns the contents of a given element in an SDF list
SUBLIST-NUMBER( )	Returns the number of elements contained in an SDF list
SUBSTRING( )	Extracts a substring from the specified string
SYSCMD( )	Returns the name of the file assigned to the system file SYSCMD
SYSDTA( )	Returns the name of the file assigned to the system file SYSDTA
SYS-ID( )	Returns the system ID
SYSLST( )	Returns the name of the file assigned to the system file SYSLST

Function	Brief description
SYSOUT( )	Returns the name of the file assigned to the system file SYSOUT
SYSTEM-CALL( )	Returns information on the syntax file containing the implemented command
SYSTEM-INFORMATION( )	Returns information on system parameters and class 2 options
TASK-MODE( )	Returns the operating mode of the current task
TIME( )	Returns the current time of day
TO-C-LITERAL( )	Converts the specified string to a C literal
TO-X-LITERAL( )	Converts the hexadecimal value of a string to its external representation
TRANSLATE( )	Allocates freely selectable results to freely selectable input values
TRANSLATE-BOOLEAN( )	Checks whether an expression is true or false
TRIM( )	Returns the TSN (job number) of the current task
TSN( )	Converts all lowercase letters in the specified string to uppercase
UPPER-CASE( )	Returns the user ID for the current task
USER-IDENTIFICATION( )	Returns the status of the specified user switch
USER-SWITCH( )	Returns the value of the specified variable attribute
VARIABLE-ATTRIBUTE( )	Converts an S variable of the type STRUCTURE to a string
VARIABLE-TO-STRING( )	Compares two strings, returning the position of the first character in string (STRING) that does not exist in search string (PATTERN)
VERIFY( )	Compares a string with a particular pattern
WILDCARD( )	Returns the operating mode of the current task
X-LITERAL-TO-INTEGER( )	Converts a string which is up to 4 bytes long to an integer (inverse function to INTEGER-TO-X-LITERAL)

Functions

ACCOUNT( ) Query account number

ACCOUNT( ) determines the account number of the current task which was specified in the SET-LOGON-PARAMETERS command.

Result type: STRING (<string 1 .. 8>)

ACCOUNT( )

ARRAY-INDEX( ) Query array index

The ARRAY-INDEX( ) function can be applied to arrays. ARRAY-INDEX supplies the value of an array index. As a result, other functions can then explicitly request this element via the array index.

Result type: INTEGER

ARRAY-INDEX( )
ARRAY-NAME = string_expression ,INDEX = *FIRST / *LAST / *LOWER-BOUND / *UPPER-BOUND

BOOLEAN( ) Convert to Boolean value

The BOOLEAN( ) function converts the expression specified in the function call to a BOOLEAN value.

Result type: BOOLEAN

BOOLEAN( ) BOOLE( )
EXPRESSION = expression

CHARACTER-TO-INTEGER( ) Convert character to integer

The CHARACTER-TO-INTEGER( ) function converts one character to a decimal number based on the characters EBCDIC code.

If the input string consists of several characters, then only the first character is converted.

All characters in a string can be converted in combination with the corresponding string function (e.g. SUBSTRING).

Result type: INTEGER (<integer 0..255>)

CHARACTER-TO-INTEGER( ) CHAR-TO-INT( )
STRING = string_expression

CHECK-DATA-TYPE( ) Check operand value

CHECK-DATA-TYPE( ) checks the data type of strings or operand values to determine whether they satisfy SDF data type requirements.

Result type: BOOLEAN

DATA-TYPE=	Valid operand combinations
*NOCHECK	VALUE, PATTERN
*INTEGER	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*X-STRING	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH, ODD
*C-STRING	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*NAME	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH, UNDERSCORE
*ALPHANUMERIC-NAME	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*STRUCTURED-NAME	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*FILENAME *PARTIAL-FILENAME *POSIX-FILENAME *POSIX-PATHNAME	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH, PATTERN, CAT-ID, USER-ID, VERSION, GENERATION, WILDCARD
*TIME	VALUE
*DATE	VALUE
*COMPOSED-NAME	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*TEXT	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*CAT-ID	VALUE
*KEYWORD	VALUE, KEYSTAR
*KEYWORD-NUMBER	VALUE, KEYSTAR
*VSN	VALUE
*X-TEXT	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH
*FIXED	VALUE, SHORTEST-LENGTH, LONGEST-LENGTH, DECIMAL-DIGITS-SHORTEST, DECIMAL-DIGITS-LONGEST
*DEVICE	VALUE, ALIAS, VOLUME-ONLY, DEVICE-CLASS, EXCEPT-DISKS, EXCEPT-TAPES
*PRODUCT-VERSION	VALUE, CORRECTION-STATE, USER-INTERFACE

CHECK-DATA-TYPE( )
INPUT = string_expression ,DATA-TYPE = *NOCHECK / *INTEGER / *X-STRING / *C-STRING / *NAME / *ALPHANUMERIC-NAME/ *STRUCTURED-NAME / *FILENAME / *FULL-FILENAME / *PARTIAL-FILENAME / *POSIX-FILENAME / *POSIX-PATHNAME / *TIME / *DATE / *COMPOSED-NAME / *TEXT / *CAT-ID / *KEYWORD / *KEYWORD-NUMBER / *VSN / *X-TEXT / *FIXED / *DEVICE / *PRODUCT-VERSION ,SHORTEST-LENGTH = *ANY / arithm_expression ,LONGEST-LENGTH = *ANY / arithm_expression ,LONGEST-LOGICAL-LENGTH = *NONE / arithm_expression ,DECIMAL-DIGITS-SHORTEST = 0 / arithm_expression ,DECIMAL-DIGITS-LONGEST = 0 / arithm_expression ,VALUE = *NO / list-poss: string_expression ,PATTERN = *NO / string_expression ,CAT-ID = *YES / *NO ,USER-ID = *YES / *NO ,VERSION = *YES / *NO ,GENERATION = *YES / *NO ,WILDCARD = *NO / *YES ,KEYSTAR = *NO / *YES ,SEPARATORS = *YES / *NO ,UNDERSCORE = *NO / *YES ,ODD = *YES / *NO ,CORRECTION-STATE = *YES / *NO / *ANY ,USER-INTERFACE = *YES / *NO / *ANY ,ALIAS = *YES / *NO ,VOLUME-ONLY = *NO / *YES ,WILDCARD-TYPE = *SELECTOR / *CONSTRUCTOR ,LOWER-CASE = *NO / *YES ,QUOTES = *OPTIONAL / *MANDATORY ,TEMPORARY-FILE = *YES / *NO ,SCOPE = *ALL / *STD-DISK ,DEVICE-CLASS = *DISK / *TAPE / *DISK-OR-TAPE ,EXCEPT-DISKS = *NONE / list-poss: string_expression ,EXCEPT-TAPES = *NONE / list-poss: string_expression

COUNTER( ) Count function calls

The COUNTER( ) function keeps track of the number of times COUNTER() is called in the current task. The counter is incremented by 1 each time COUNTER() is called.

Result type: INTEGER (<integer 1 .. 2147483647>)

COUNTER( )

CURRENT-TYPE( ) Query variable type

The CURRENT-TYPE( ) function returns the current type of the value of a simple variable (this must not be confused with the current type of a variable declaration, which is returned by the VARIABLE-ATTRIBUTE( ) function). If the variable type has not yet been defined (TYPE = *ANY), or if CURRENT-TYPE( ) is applied to a complex variable, *NONE is returned as the result.

Result type: STRING

CURRENT-TYPE( ) CURR-TYPE( )
VARIABLE-NAME = string_expression

DATE( ) Output date

The DATE( ) function determines the current date and returns it in the specified format.

Result type: STRING (<string 10..13>)

DATE( )
FORMAT = *ISO / *AMERICAN / *GERMAN ,MODE = *LOCAL-TIME / *UNIVERSAL-TIME

DATE-VALUE( ) Output particular date

The DATE-VALUE( ) function outputs the date which is a specified number of days from the base date (default value for this is the start of the 20th century (1900-01-01)).

Result type: STRING (<string 10..13>

DATE-VALUE( )
NUMBER-OF-DAYS = arithm_expression ,BASE = <u>*STD</u> / *TODAY / string_expression ,FORMAT = <u>*ISO</u> / *AMERICAN / *GERMAN

DAY( ) Output day of the week

The DAY( ) function supplies the name of the current day of the week in the specified language, but only in abbreviated form.

Result type: STRING (<string 2..3>)

DAY( )
LANGUAGE = <u>*ENGLISH</u> / *GERMAN / *STD

ELAPSED-DAYS( ) Output number of days difference

The ELAPSED-DAYS( ) function outputs the number of days difference between two specified dates.

Result type: INTEGER (<integer -3074323 .. 3074323>)

ELAPSED-DAYS( )
DATE = string_expression ,BASE = <u>*STD</u> / *TODAY / string_expression

EXPLICIT-CALL( ) Output explicit command call

EXPLICIT-CALL( ) outputs the type of call to a procedure. Here, TRUE means that the call was explicit (i.e. by a CALL-PROCEDURE, INCLUDE-PROCEDURE) and FALSE means an implicit call (e.g. when the call is made by a command in a procedure).

Result type: BOOLEAN

EXPLICIT-CALL( )

EXTEND-SDF-LIST( ) Append list element

The EXTEND-SDF-LIST( ) function appends a new element to an SDF list. This new element may itself be an SDF list.

Result type: STRING

EXTEND-SDF-LIST( )
LIST = string_expression ,ELEMENT = string_expression ,POSITION = <u>*LAST</u> / *FIRST / arithm_expression

EXTRACT-FIELD( ) Extract field

The EXTRACT-FIELD( ) function extracts a field from an input string.

Result type: STRING

EXTRACT-FIELD( )
STRING = string_expression ,FIELD-NUMBER = arithm_expression ,FIELD-SEPARATOR = <u>*ANY-BLANKS</u> / string_expression

FILL( ) Fill string

The FILL( ) function fills the string specified in the function call with zeros up to the specified length and in the specified direction. This fill character can be defined in the function call.

Result type: STRING

FILL( )
STRING = string_expression
,LENGTH = arithm_expression
,SIDE = *RIGHT / *LEFT
,FILL-BYTE = C'␣' / character

FIRST-VARIABLE-NAME( ) Request variable element name

The FIRST-VARIABLE-NAME( ) function can be used to analyze the format of complex variables, primarily in combination with the NEXT-VARIABLE-NAME( ) function. FIRST-VARIABLE-NAME( ) can be applied to all aggregates and lists. FIRST-VARIABLE-NAME( ) begins with the specified variable or variable element name and then supplies the name of the first variable. If there is no lower level, FIRST-VARIABLE-NAME( ) returns *END.

Result type: STRING (<composed-name 1..255> or '*END')

FIRST-VARIABLE-NAME( ) FIRST-VAR-NAME( )
VARIABLE-NAME = string_expression

FROM-C-LITERAL( ) Convert C literal

The FROM-C-LITERAL( ) function converts a C literal to the corresponding string value. The leading C and the single quotes at the beginning and end of the literal are deleted.

FROM-C-LITERAL( ) is the reverse function of TO-C-LITERAL( ).

Result type: STRING

FROM-C-LITERAL( ) FROM-C-LIT( )
STRING = string_expression

FROM-X-LITERAL( ) Convert X literal

The FROM-X-LITERAL( ) function converts an X literal to the corresponding string value.

FROM-X-LITERAL( ) is the reverse function of TO-X-LITERAL( ).

Result type: STRING

FROM-X-LITERAL( ) FROM-X-LIT( )
STRING = string_expression

HASH-STRING( ) Encrypt expression as string

The HASH-STRING( ) function converts a string expression to a string with a binary content with any required length (e.g. a password). The algorithm which is used has a high probability of returning different output values for different input strings.

Result type: STRING

HASH-STRING( )
STRING = string_expression ,LENGTH = 4 / arithm_expression

HASH-VALUE( ) Encrypt expression as integer value

The HASH-VALUE( ) function converts a string expression to an integer value; the algorithm which is used to do this has a high probability of returning different output values for different input strings.

Result type: INTEGER (<integer $-2^{31}..2^{31}-1$>)

HASH-VALUE( )
STRING = string_expression

HOME-CAT-ID( ) Request catalog ID of home pubset

The HOME-CAT-ID( ) function supplies the catalog ID of the home pubset.
The catalog ID is assigned by the system administrator and can be up to four characters long.
The home pubset of the running system is the pubset from which the system was loaded.

Result type: STRING (<string 1..4>)

HOME-CAT-ID( )

HOST( ) Query host name

The HOST( ) function supplies the internal name of the host on which the function is called.
The system administrator defines this internal name when DCM is started.

Result type: STRING (<string 1..8>)

HOST( )

INDEX( ) Search for string

The INDEX( ) function indicates the position of a search string within the overall string. The overall string can be searched from left to right or from right to left; the result value always relates to the beginning of the overall string.

Result type: INTEGER

INDEX( )
STRING = string_expression$_1$,PATTERN = string_expression$_2$,DIRECTION = *FORWARD / *REVERSE ,BEGIN-COLUMN = 1 / arithm_expression ,END-COLUMN = *LAST / arithm_expression

INSTALLATION-PATH( ) Output path name

The INSTALLATION-PATH( ) specifies the path name assigned from the SCI for the logical name of a file (installation item) that belongs to a specific product version.

Result type: STRING

INSTALLATION-PATH( )
LOGICAL-ID = string_expression ,INSTALLATION-UNIT = string_expression ,VERSION = *STD / string_expression ,DEFAULT-PATH-NAME = string_expression

INTEGER( ) Convert expression to integer

INTEGER( ) converts any expression to the data type INTEGER

Result type: INTEGER

INTEGER( ) INT( )
EXPRESSION = expression

INTEGER-TO-CHARACTER( ) Convert integer to character

The INTEGER-TO-CHARACTER( ) function converts an integer into a character (C string).

The number is interpreted as the integer value of the EBCDI code for a character, and this character is returned.

INTEGER-TO-CHARACTER( ) is the reverse function of CHARACTER-TO-INTEGER( ).

Result type: STRING (<string 1..1>)

INTEGER-TO-CHARACTER( ) INT-TO-CHAR( )
INTEGER = arithm_expression

INTEGER-TO-X-LITERAL( ) Convert integer to X literal

Converts an integer to an X literal containing the 4-byte long coding of the integer.

INTEGER-TO-X-LITERAL( ) is the inverse function to X-LITERAL-TO-INTEGER( ).

Result type: STRING

INTEGER-TO-X-LITERAL( ) INT-TO-X-LIT( )
STRING = string_expression

IS-C-LITERAL( ) Check C literal

The IS-C-LITERAL( ) function checks whether the specified string is a C literal and can be converted to a string (using the FROM-C-LITERAL function).

Result type: BOOLEAN

IS-C-LITERAL( ) IS-C-LIT( )
STRING = string_expression

IS-CATALOGED-FILE( ) Check catalog entry

The IS-CATALOGED-FILE( ) function checks whether there is a catalog entry with the specified file name. The response when an error occurs (invalid file name, etc.) can be defined.

Result type: BOOLEAN

IS-CATALOGED-FILE( ) IS-CAT-FILE( )
FILE = string_expression ,ERROR-REPORTING = <u>*PROC-ERROR-MECHANISM</u> / *RETURN-FALSE ,ERROR-VARIABLE = <u>*NONE</u> / string_expression

IS-CATALOGED-JV( ) Request job variable

The IS-CATALOGED-JV( ) function checks whether a catalog entry exists for the specified job variable names, i.e. whether the specified job variable exists.

Result type: BOOLEAN

IS-CATALOGED-JV( ) IS-CAT-JV( )
JV = string_expression ,ERROR-REPORTING = <u>*PROC-ERROR-MECHANISM</u> / *RETURN-FALSE ,ERROR-VARIABLE = <u>*NONE</u> / string_expression

IS-DECLARED( ) Check variable declaration

The IS-DECLARED( ) function checks whether the specified simple or complex variable has already been declared.

Result type: BOOLEAN

IS-DECLARED( )
VARIABLE-NAME = string_expression ,SCOPE = <u>*BY-HIERARCHY</u> / *TASK / *CALLING-PROCEDURES

IS-EMPTY-FILE( ) Check file size

The IS-EMPTY-FILE( ) function checks whether a file is empty (last-page pointer points to page 0). This is true if the output field HIGH-US-PA in the output from the SHOW-FILE-ATTRIBUTES command contains the value 0.

Result type: BOOLEAN

IS-EMPTY-FILE( )
FILE-NAME = string_expression

IS-INITIALIZED( ) Check variable initialization

The IS-INITIALIZED( ) function checks whether the specified variable is initialized, i.e. whether its content is valid. Even the null string is a valid variable content.

A variable can be initialized only if it has been declared.
Only simple variables or list variables can be checked.

Result type: BOOLEAN

IS-INITIALIZED( )
VARIABLE-NAME = string_expression

IS-INTEGER( ) Check expression

IS-INTEGER( ) checks whether the expression specified as a string represents an integer.

Result type: BOOLEAN

IS-INTEGER( )
STRING = string_expression

IS-LIBRARY( ) Check library name

The IS-LIBRARY( ) function checks whether the specified file is entered in the catalog as a PLAM library.

If the file is not entered in the catalog as a PLAM library, IS-LIBRARY( ) supplies the result FALSE.

Result type: BOOLEAN

IS-LIBRARY( )
FILE = string_expression

IS-LIBRARY-ELEMENT( ) Check library element

The IS-LIBRARY-ELEMENT( ) function checks whether or not the specified library element exists.

Result type: BOOLEAN

IS-LIBRARY-ELEMENT( ) IS-LIB-ELEM( )
LIBRARY = string_expression ,ELEMENT = string_expression ,TYPE = string_expression ,VERSION = <u>*HIGHEST-EXISTING</u> / string_expression

IS-SDF-LIST( ) Analyze string against criteria for SDF lists

The IS-SDF-LIST( ) function analyzes whether a string expression is an SDF list of the format '<element1>,<element2>,...,<elementn>)' where <elementi> is a character sequence that must not contain any comma (except within parentheses).

Result type: BOOLEAN

IS-SDF-LIST( )
STRING = string_expression

IS-SDF-P( ) Check whether SDF-P is loaded

The IS-SDF-P( ) function checks whether SDF-P is loaded in the system. If it is loaded, the result TRUE is returned.

Result type: BOOLEAN

IS-SDF-P( )

IS-SDF-STRUCTURE( ) Analyze string against criteria for SDF structures

The IS-SDF-STRUCTURE( ) function analyzes whether the specified string is an SDF structure.

The specified SDF structure must be introduced by a value. This value must not be omitted, or else the string will be regarded as an SDF list and not as an SDF structure.

Result type: BOOLEAN

IS-SDF-STRUCTURE( )
STRING = string_expression

IS-VARIABLE-NAME( ) Check variable name

The IS-VARIABLE-NAME( ) function checks whether the specified string is a syntactically correct variable name. This check is a pure syntax check. It does not check whether a variable with this name exists.

Result type: BOOLEAN

IS-VARIABLE-NAME( ) IS-VAR-NAME( )
STRING = string_expression

IS-X-LITERAL( ) Check X literal

The IS-X-LITERAL( ) function checks whether the specified string expression contains an X literal and can be converted with FROM-X-LITERAL( ).

Result type: BOOLEAN

IS-X-LITERAL( ) IS-X-LIT( )
STRING = string_expression

JOB-CLASS( ) Request job class

The JOB-CLASS( ) function requests the job class to which the current task belongs

Result type: STRING

JOB-CLASS( )

JOB-MONJV( ) Request MONJV

The JOB-MONJV( ) function supplies the name of the job variable which monitors the job.

Result type: STRING

JOB-MONJV( )

JOB-NAME( ) Request job name

The JOB-NAME( ) function supplies the job name of the current task, i.e. the name specified in the SET-LOGON-PARAMETERS command.

Result type: STRING (<string 1..8>)

JOB-NAME( )

JV( ) Request job variable

The JV( ) function supplies the content of the specified job variable.

Result type: STRING

JV( )
JV-NAME = string_expression ,START = <u>1</u> / arithm_expression$_1$,LENGTH = <u>*REST-LENGTH</u> / arithm_expression$_2$

LAYOUT-SCOPE( ) Request layout scope

The LAYOUT-SCOPE( ) function applies only to structure layouts and supplies their scope.

Result type: STRING

LAYOUT-SCOPE( )
LAYOUT-NAME = string_expression

LENGTH( ) Request string length

The LENGTH( ) function supplies the length of the specified string.

Result type: INTEGER

LENGTH( )
STRING = string_expression

LIMIT( ) Request maximum list size

The LIMIT( ) function can be applied only to lists. LIMIT( ) requests the number of elements which a list variable can contain.

Result type: INTEGER (<integer 1 .. 2147483647>)

LIMIT( )
LIST-NAME = string_expression

LOGGING-MODE( ) Check logging

The LOGGING-MODE( ) function indicates whether the logging function was activated when calling the CALL-PROCEDURE or INCLUDE-PROCEDURE command.

Result type: STRING ('YES' / 'NO')

LOGGING-MODE( ) LOG-MODE( )
STREAM = *CMD / *DATA

LOWER-CASE( ) Convert uppercase letters to lowercase

The LOWER-CASE( ) function converts all uppercase letters in the specified string to lowercase.

The letters which are to be converted must correspond to the standard EBCDI code.

Result type: STRING

LOWER-CASE( )
STRING =string_expression ,TRANSLATE = <u>*ALL</u> / *OUTSIDE-QUOTES-ONLY / *INSIDE-QUOTES-ONLY

MAINCODE( ) Request error code

The MAINCODE( ) function accesses the return code of the last command which resulted in an error or which was followed by a /SAVE-RETURNCODE. It returns the seven-byte error code of the return code, which is also the message code for error messages (the remaining components of the command return code are requested with the SUBCODE1( ) and SUBCODE2( ) functions).

Result type: STRING (<string 7..7>)

MAINCODE( ) MC( )

MONTH( ) Output name of month

The MONTH( ) function supplies the name of the current month in the specified language and in the form of a three-character abbreviation.

Result type: STRING (<string 1..3>)

MONTH( )
LANGUAGE = <u>*ENGLISH</u> / *GERMAN / *STD

MSG( ) Output message text

The MSG( ) function supplies the message text assigned to the specified message code; this is done in the specified language and in the specified output format.

Result type: STRING (<string>)

MSG( )
MSG-IDENTIFICATION = string_expression ,LANGUAGE = *STD / *ENGLISH / *GERMAN ,INSERT-00 = *NONE / string_expression ,INSERT-01 = *NONE / string_expression : : : ,INSERT-29 = *NONE / string_expression ,MSG-STRUCTURE-OUTPUT = *NONE / *SYSMSG

NEXT-VARIABLE-NAME( ) Request variable level

The NEXT-VARIABLE-NAME( ) function can be used to analyze the layout of complex variables, primarily in connection with the FIRST-VARIABLE-NAME() function.

The NEXT-VARIABLE-NAME( ) function supplies the name of the next variable element on the same level. If there are no more variable elements on this level, NEXT-VARIABLE-NAME( ) returns *END.

Result type: STRING

NEXT-VARIABLE-NAME( ) NEXT-VAR-NAME( )
VARIABLE-NAME = string_expression

PROC-LEVEL( ) Request nesting level

The PROC-LEVEL( ) function supplies the current nesting level of the S procedure.

Result type: INTEGER

PROC-LEVEL( )

PROCESSOR( ) Request processor name

The PROCESSOR( ) function supplies the processor name of the TIAM station, i.e. the terminal at which the current task was started.

Result type: STRING (<string 1..8>)

PROCESSOR( )

PROG-MONJV( ) Request MONJV program

The PROG-MONJV( ) function supplies the name of the job variable which monitors the loaded program.

Result type: STRING (<string 1..255>)

PROG-MONJV( )

PROG-NAME( ) Request program name

The PROG-NAME( ) function supplies the internal name (truncated to eight characters) of the currently loaded program.

Result type: STRING

PROG-NAME( )

RENAME( ) Generate new name using wildcards

The RENAME( ) function provides a new name. This new name is generated on the basis of the input name, using wildcards.

Result type: STRING

RENAME( )
INPUT-NAME = string_expression ,WILDCARD-PATTERN = string_expression ,CONSTRUCTION-WILDCARD = string_expression ,NO-MATCH = *WARNING / *IGNORE / *ERROR ,WILDCARD-MODE = *BS2000 / *POSIX

REPLACE( ) Overwrite or replace substring

The REPLACE( ) function overwrites or replaces a substring within a string with another string. This can make the original string longer.

Result type: STRING

REPLACE( )
STRING = string_expression$_1$,START = 1 / zahl ,REPLACE = string_expression$_2$,SUPPRESSED-LENGTH = *REPLACE-LENGTH / zahl

RUN-PRIORITY( ) Request runtime priority

The RUN-PRIORITY( ) function supplies the priority level of the current job.

The supplied value can then be checked and - if necessary - the priority of the task changed.

Result type: INTEGER (<integer 0..255>)

RUN-PRIORITY( ) RUN-PRIO( )

SDF-P-VERSION( ) Request SDF-P version

The SDF-P-VERSION( ) function supplies information about the installed version of (chargeable) subsystem SDF-P or about the current version of the SDF-P-BASYS subsystem.

Result type: STRING

SDF-P-VERSION( )
FUNCTION-RANGE = *STD / *BASIC

SDF-STRUCTURE-VALUE( ) Output value of structure

The SDF-STRUCTURE-VALUE( ) function supplies part or all of the contents of an SDF structure.

Result type: STRING (<string>)

SDF-STRUCTURE-VALUE( )
STRING = string_expression
,OPERAND-NAME = *ROOT / string_expression / arithm_expression
,ATTACHED-STRUCTURE = *YES / *NO

SEARCH-LIST-INDEX( ) Search for string in list

The function SEARCH-LIST-INDEX( ) searches a list variable for a string or for a regular expression which has been formed in accordance with POSIX rules. The return value indicates the number of the first list element containing this expression or search string.

Result type: INTEGER

SEARCH-LIST-INDEX( )
LIST-VARIABLE-NAME = string_expression$_1$
,PATTERN = string_expression$_2$
,BEGIN-INDEX = 1 / arithm_expression
,END-INDEX = *LAST / arithm_expression
,BEGIN-COLUMN = 1 / arithm_expression
,END-COLUMN = *LAST / arithm_expression
,PATTERN-TYPE = *STRING / *REGULAR-EXPRESSION
,DIRECTION = *FORWARD / *REVERSE

SESSION-NUMBER( ) Request system sequence number

The SESSION-NUMBER( ) function supplies the system sequence number of the system currently running (for example, the system sequence number can be part of the CONSLOG file name).

Result type: STRING (<string 3..3>)

SESSION-NUMBER( )

SIZE( ) Request size of complex variables

SIZE( ) requests the number of elements comprising the specified variable.

Result type: INTEGER

SIZE( )
VARIABLE-NAME = string_expression

STATION( ) Request TIAM station name

The STATION( ) function supplies the station name of the TIAM station from which the procedure was initiated.

Result type: STRING(<string 1..8>)

STATION( )

STATION-TYPE( ) Request TIAM device type

The STATION-TYPE( ) function returns the generated device type of the TIAM station from which the procedure was called.

Result type: STRING(<string 1..8>)

STATION-TYPE( )

STD-CAT-ID( ) Request catalog ID

The STD-CAT-ID( ) function supplies the ID of the pubset assigned to the current user ID as the default pubset.

Result type: STRING (<string 1..4>)

STD-CAT-ID( )

STMT-SPINOFF( ) Request statement spin-off

The STMT-SPINOFF( ) function indicates whether a statement spin-off has been activated for the loaded program.

Result type: STRING (YES / NO / UNDEFINED)

STMT-SPINOFF( )

STRING( ) Convert expression to string

The STRING( ) function converts an INTEGER, BOOLEAN or STRING expression to data type STRING. The rules for implicit conversion apply.

Result type: STRING

STRING( ) STR( )
EXPRESSION = expression

SUBCODE1( ) Request subcode1

The SUBCODE1( ) function supplies the error class of the current command return code, i.e. the return code of the last command which resulted in an error or which was followed by the /SAVE-RETURNCODE command.

Result type: INTEGER (<integer 0..255>)

SUBCODE1( ) SC1( )

SUBCODE2( ) Request subcode2

The SUBCODE2( ) function supplies the severity of the current command return code, i.e. the return code of the last command which resulted in an error or which was followed by /SAVE-RETURNCODE.

Result type: INTEGER (<integer 0..255>)

SUBCODE2( ) SC2( )

SUBLIST( ) Select element from SDF list

The SUBLIST( ) function returns the contents of the selected element of an SDF list. An SDF list is a string which is interpreted in accordance with the syntactical rules for operand lists in commands. Evaluation of the string with this function is meaningful only if it has the format ’(<element$_1$>,<element$_2$>,...<elementn>)’, where <element$_i$> is a sequence of characters which contain no commas outside pairs of parentheses.
The IS-SDF-LIST() function can be used to check whether a string is a list.

Result type: STRING

SUBLIST( )
LIST = string_expression ,POSITION = arithm_expression

SUBLIST-NUMBER( ) Request number of elements in SDF list

The SUBLIST-NUMBER( ) function provides information on how many elements an SDF list contains. An SDF list is a string which is interpreted in accordance with the syntactical rules for operand lists in commands. Evaluation of the string with this function is meaningful only if it has the format '(<element$_1$>,<element$_2$>,...<element$_n$>)', where <element$_i$> is a sequence of characters which contain no commas outside pairs of parentheses.
The IS-SDF-LIST() function can be used to check whether a string is a list.

Result type: INTEGER

SUBLIST-NUMBER( )
LIST = string_expression

SUBSTRING( ) Output substring

The SUBSTRING( ) function extracts a substring from the specified string. The start position of the substring and its length are determined by means of the input parameters.

Result type: STRING

SUBSTRING( ) SUBSTR( )
STRING = string_expression ,START = <u>1</u> / arithm_expression$_1$,LENGTH = <u>*REST-LENGTH</u> / arithm_expression$_2$

SYSCMD( ) Request SYSCMD assignment

The SYSCMD( ) function supplies the name of the file (alternative: a library element or a list variable) assigned to the system file SYSCMD.

Result type: STRING

SYSCMD( )
SYSTEM-FILE-CONTEXT = <u>*OWN</u> / *CALLER

SYSDTA( ) Request SYSDTA assignment

The SYSDTA( ) function supplies the name of the file (alternative: a library element or a list variable) assigned to the system file SYSDTA.

Result type: STRING

SYSDTA( )

SYS-ID( ) Request system identification

The SYS-ID( ) function supplies the system identification, i.e. the system ID of the current system.

Result type: STRING (<string 1..4>)

SYS-ID( )

SYSLST( ) Request SYSLST assignment

The SYSLST( ) function supplies the name of the file (alternative: a library element or a list variable) assigned to the system file SYSLST.

Result type: STRING

SYSLST( )

SYSOUT( ) Request SYSOUT assignment

The SYSOUT( ) function supplies the name of the file (alternative: a library element or a list variable) assigned to the system file SYSOUT.

Result type: STRING

.

SYSOUT( )

SYSTEM-CALL( ) Output command source

SYSTEM-CALL( ) is used within a procedure which has been called as the implementor of a command. The functinon determines the source of the command, i.e. whether it originates from the system syntax file or the group syntax file.

Result type: BOOLEAN

SYSTEM-CALL( )

SYSTEM-INFORMATION( ) Request system information

The SYSTEM-INFORMATION( ) function can be used to request system information and system parameters. One value can be queried per call.

Restrictions

This function is equivalent to the Executive macro SINF at program level. The SINF macro will not be developed further and has been replaced by the NSIINF and NSIOPT macros. System information and system parameters which were introduced only after this macro was replaced **cannot** be queried with the SYSTEM-INFORMATION function.
All the system information and system parameters which currently exist can be queried using the SHOW-SYSTEM-INFORMATION and SHOW-SYSTEM-PARAMETERS commands (these commands also support structured output to S variables).

Result type: STRING

SYSTEM-INFORMATION( ) SYS-INF( )
INFORMATION = string_expression

TASK-MODE( ) Request task mode

Supplies the mode of the current task.

Result type: STRING (<string 2..6>)

TASK-MODE( )

TIME( ) Request time

The TIME( ) function supplies the current time of day; the separator between the entries for hours, minutes and seconds can be freely selected.

Result type: STRING (<string 8..8>)

TIME( )
SEPARATOR = ':' / character ,MODE = *LOCAL-TIME / *UNIVERSAL-TIME

TO-C-LITERAL( ) Convert string to C literal

The TO-C-LITERAL( ) function converts the specified string to a C literal by placing single quotes at the beginning and end of the string and doubling single quotes within the string. The FROM-C-LITERAL( ) function is the inverse of the TO-C-LITERAL( ) function.

Result type: STRING

TO-C-LITERAL( ) TO-C-LIT( )
STRING = string_expression

TO-X-LITERAL( ) Convert string to X literal

The TO-X-LITERAL( ) function converts the hexadecimal value of the specified string into the external representation: it places single quotes at the beginning and end of the string and also places an X in front of the string.

The FROM-X-LITERAL( ) function is the inverse of the TO-X-LITERAL( ) function.

Result type: STRING

TO-X-LITERAL( ) TO-X-LIT( )
STRING = string_expression

TRANSLATE( ) Assign result values to input values

The TRANSLATE( ) function can be used to assign any result data to any input data. Up to ten translation operations can be specified; WHEN and THEN clauses must be present in pairs.

Result type: STRING

TRANSLATE( )
STRING = string_expression$_0$,WHEN1 = <u>*NONE</u> / string_expression$_1$, THEN1 = <u>*NONE</u> / *SAME / string_expression$_{11}$,WHEN2 = <u>*NONE</u> / string_expression$_2$, THEN2 = <u>*NONE</u> / *SAME / string_expression$_{12}$,WHEN3 = <u>*NONE</u> / string_expression$_3$, THEN3 = <u>*NONE</u> / *SAME / string_expression$_{13}$,WHEN4 = <u>*NONE</u> / string_expression$_4$, THEN4 = <u>*NONE</u> / *SAME / string_expression$_{14}$,WHEN5 = <u>*NONE</u> / string_expression$_5$, THEN5 = <u>*NONE</u> / *SAME / string_expression$_{15}$,WHEN6 = <u>*NONE</u> / string_expression$_6$, THEN6 = <u>*NONE</u> / *SAME / string_expression$_{16}$,WHEN7 = <u>*NONE</u> / string_expression$_7$, THEN7 = <u>*NONE</u> / *SAME / string_expression$_{17}$,WHEN8 = <u>*NONE</u> / string_expression$_8$, THEN8 = <u>*NONE</u> / *SAME / string_expression$_{18}$,WHEN9 = <u>*NONE</u> / string_expression$_9$, THEN9 = <u>*NONE</u> / *SAME / string_expression$_{19}$,WHEN10 = <u>*NONE</u> / string_expression$_{10}$, THEN10 = <u>*NONE</u> / *SAME / string_expression$_{20}$,ELSE = <u>*NONE</u> / *SAME / string_expression$_{21}$

TRANSLATE-BOOLEAN( ) Check Boolean expression

The TRANSLATE-BOOLEAN( ) function checks whether the input expression is true or false. If it is true, the value specified in the THEN clause is supplied as the result. If the input expression is not true (= false), the value specified in the ELSE clause is supplied.

Result type: BOOLEAN / INTEGER / STRING

TRANSLATE-BOOLEAN( )
IF = expression$_1$,THEN = expression$_2$,ELSE = expression$_3$

TRIM( ) Remove matching characters at the beginning or end of a string

The TRIM( ) function removes matching characters at the beginning, end or both ends of a string.

Result type: STRING

TRIM( )
STRING = string_expression ,SIDE = *BOTH / *LEFT / *RIGHT ,TRIM-BYTE = C'␣' / character

TSN( ) Request TSN

The TSN( ) function supplies the task sequence number of the current job.

Result type: STRING (<string 4..4>)

TSN( )

UPPER-CASE( ) Convert lowercase letters into uppercase

The UPPER-CASE( ) function converts all lowercase letters in the specified string into uppercase letters.

The letters which are to be converted must correspond to the standard EBCDI code. There is no support for language-specific variants.

Result type: STRING

UPPER-CASE( )
STRING =string_expression ,TRANSLATE = *ALL / *OUTSIDE-QUOTES-ONLY / *INSIDE-QUOTES-ONLY

USER-IDENTIFICATION( ) Request user identification

The USER-IDENTIFICATION( ) function supplies the user identification of the current job, i.e. the user identification from the SET-LOGON-PARAMETERS command.

Result type: STRING (<string 1..8>)

USER-IDENTIFICATION( ) USER-ID( )

USER-SWITCH( ) Evaluate user switch

The USER-SWITCH( ) function checks the value of the specified user switch.

User switches are used, for example, to synchronize batch jobs, i.e. in background procedures. Each user ID has available to it 32 user switches, which apply for all the jobs running under the user ID. That is, user switches which are set in one job can be evaluated by a different job which is running under the same user ID. User switches are set by means of the MODIFY-USER-SWITCHES command.

Result type: BOOLEAN

USER-SWITCH( )
NUMBER = zahl ,USER-ID = *OWN / <string 1..8>

VARIABLE-ATTRIBUTE( ) Request variable attributes

VARIABLE-ATTRIBUTE() returns the value of the specified attribute for the specified variable.

Result type: STRING

VARIABLE-ATTRIBUTE( ) VAR-ATTR( )
VARIABLE-NAME = string_expression ,ATTRIBUTE = *TYPE / *CONTAINER / *CONTAINER-NAME / *CONTAINER-SCOPE / *MULTIPLE-ELEMENTS / *SCOPE / *STRUCTURE-INFO

VARIABLE-TO-STRING( ) Convert variable

The VARIABLE-TO-STRING( ) function converts an S variable of type structure into a string (SDF command string).

Result type: STRING

VARIABLE-TO-STRING( ) VAR-TO-STR( )
VARIABLE-NAME = string_expression

VERIFY( ) Verify strings

The VERIFY( ) function compares two strings and returns the position of the first character in the string (STRING) which is not contained in the search string (PATTERN). The number and the order of the characters in the two strings are ignored. The search direction can be freely selected; the returned position value is always with respect to the beginning of the first string.

Result type: INTEGER

VERIFY( )
STRING = $string\_expression_1$,PATTERN = $string\_expression_2$,DIRECTION = <u>*FORWARD</u> / *REVERSE

WILDCARD( ) Search for pattern

The WILDCARD( ) function compares a string with a pattern string. The pattern string is subject to the general rules for patterns in BS2000.

Result type: BOOLEAN

WILDCARD( )
STRING = string_expression1 ,PATTERN = string_expression2 ,WILDCARD-MODE = <u>*BS2000</u> / *POSIX

X-LITERAL-TO-INTEGER( ) Convert string to integer

Converts a string which is up to 4 bytes long to an integer. The input string can be specified as an X string or as a C string. An empty string is assigned the value 0.
If the input string consists of less than 4 characters, it is padded from left to right with X’00’.

X-LITERAL-TO-INTEGER( ) is the inverse function to INTEGER-TO-X-LITERAL( ).

Result type: INTEGER

X-LITERAL-TO-INTEGER( ) X-LIT-TO-INT( )
STRING = string_expression

SDF-P commands

Overview

Command	Brief description
ASSIGN-STREAM	Assigns an (output) server to an S variable stream
BEGIN-BLOCK	Initiates a command block
BEGIN-PARAMETER-DECLARATION	Initiates procedure parameter declaration in the procedure head
BEGIN-STRUCTURE	Declares a static structure
CALL-PROCEDURE	Starts a command sequence
CLOSE-VARIABLE-CONTAINER	Closes variable containers
COMPILE-PROCEDURE	Compiles a procedure in the intermediate format
CYCLE	Interrupts a loop pass
DECLARE-CONSTANT	Declares a variable and assigns it a constant value
DECLARE-ELEMENT	Declares a structure element
DECLARE-PARAMETER	Declares a procedure parameter
DECLARE-VARIABLE	Declares a variable
DELETE-STREAM	Deletes an S variable stream
DELETE-VARIABLE	Deletes a variable
ELSE	Initiates the ELSE branch of an IF block
ELSE-IF	Initiates an alternative branch in an IF block
END-BLOCK	Terminates a command block
END-FOR	Terminates a FOR block
END-IF	Terminates an IF block
END-PARAMETER-DECLARATION	Terminates a procedure parameter declaration
END-STRUCTURE	Terminates a structure layout declaration
END-WHILE	Terminates a WHILE block
ENTER-PROCEDURE	Starts a procedure in the background as a batch job
EXECUTE-CMD	Controls system output to variables
EXIT-BLOCK	Terminates command block processing
EXIT-PROCEDURE	Terminates a procedure
FOR	Initiates a FOR block
FREE-VARIABLE	Deletes the contents of a variable

Command	**Brief description**
GOTO	Branches to an S tag
IF	Initiates an IF block
IF-BLOCK-ERROR	Initiates block error handling
IF-CMD-ERROR	Initiates command error handling
IMPORT-VARIABLE	Imports a variable
INCLUDE-BLOCK	Executes a BEGIN block as a subprocedure
INCLUDE-CMD	Executes a command sequence from a program
INCLUDE-PROCEDURE	Starts a command sequence as an Include procedure
MODIFY-PROCEDURE-OPTIONS	Modifies procedure attributes during procedure execution
MODIFY-PROCEDURE-TEST-OPTIONS	Modifies logging and limits the number of returns permitted
OPEN-VARIABLE-CONTAINER	Opens variable containers stored as PLAM library elements
RAISE-ERROR	Generates a return code
READ-VARIABLE	Assigns values to variables
REPEAT	Initiates a REPEAT block
REPEAT-CMD	Repeats a command
REPEAT-STMT	Repeats a statement
SAVE-RETURNCODE	Saves the current command return code
SAVE-VARIABLE-CONTAINER	Saves variable containers
SELECT-VARIABLE-ELEMENTS	Displays the elements of a variable on the screen and writes back any elements selected from these
SEND-DATA	Transfers a data record to a program
SEND-STMT	Transfers a statement record to a program
SET-PROCEDURE-OPTIONS	Defines procedure attributes
SET-VARIABLE	Assigns a value to a variable
SHOW-STREAM-ASSIGNMENT	Provides information on the current assignment of S variable streams
SHOW-STRUCTURE-LAYOUT	Provides information on the element names of a structure layout
SHOW-VARIABLE	Outputs the contents of variables
SHOW-VARIABLE-ATTRIBUTES	Provides information on the attributes of a variable (name, value, data type, ...)

Command	Brief description
SHOW-VARIABLE-CONTAINER-ATTR	Provides information on all open variable containers
SORT-VARIABLE	Sorts the elements of a list variable
TRACE-PROCEDURE	Resumes an interrupted procedure in stages
TRANSMIT-BY-STREAM	Used by the client to transfer variables from or to the server via an S variable stream
UNTIL	Terminates a REPEAT block
WHILE	Initiates a WHILE block

Privileges

Apart from a few exceptions, all commands can be called by users with the following privileges:

STD-PROCESSING
OPERATING
HARDWARE-MAINTENANCE
SECURITY-ADMINISTRATION
SAT-FILE-MANAGEMENT
SAT-FILE-EVALUATION

Exceptions

– ENTER-PROCEDURE command
necessary privilege: STD-PROCESSING or HARDWARE-MAINTENANCE

– Users with the privilege SECURITY-ADMINISTRATION, SAT-FILE-MANAGEMENT or SAT-FILE-EVALUATION can only use the following commands in procedures benutzen:

BEGIN-BLOCK
BEGIN-PARAMETER-DECLARATION
CYCLE
DECLARE-PARAMETER
ELSE
ELSE-IF
END-BLOCK
END-FOR
END-IF
END-PARAMETER-DECLARATION
END-WHILE
EXECUTE-CMD
EXIT-BLOCK
FOR
GOTO
IF
IF-BLOCK-ERROR
IF-CMD-ERROR
INCLUDE-BLOCK
INCLUDE-PROCEDURE
REPEAT
SET-PROCEDURE-OPTIONS
TRACE-PROCEDURE
UNTIL
WHILE

Command return codes

All SDF-P commands supply return codes which provide the user with information on the command execution. This command return code is comparable to the return code on the program level. The command return code allows users to respond to specifoc error situations.

The command return code consists of three parts:

– subcode1, which assigns the *error* situation to an error class indicating how serious the error is. The value of subcode 1 is output in *decimal*.
– subcode2, which can contain supplementary information on the error class.
– maincode, which corresponds to a message code and supplies specific error information. This message code can be used to output the appropriate error message via the predefined MESSAGE() function or using the SDF command HELP-MSG-INFORMATION.

The command return code can be requested with the predefined functions SUBCODE1( ), SUBCODE2( ) and MAINCODE( ).

There are separate return codes for each command. In addition to the special return codes for specific commands, there are some general return codes, which are listed below.

The general return codes (i.e. the return codes which can occur for any command) are:

(SC2)	SC1	Maincode	Meaning[1]
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No more address space available

[1] If the table also contains guaranteed messages, then "/ guaranteed messages" is added to the meaning column.

For all commands, statements and records in which expression replacement is carried out, the following return codes may appear if errors occur during replacement:

(SC2)	SC1	Maincode	Meaning
	1	SDP0140	Syntax error during replacement
	64	SDP0141	Semantic error during replacement

The following return code may appear for all data lines in a false context::

(SC2)	SC1	Maincode	Meaning
	64	SDP0091	Semantic error

The following return code may appear for all statements in a false context

(SC2)	SC1	Maincode	Meaning
	64	SDP0091	Semantic error

Guaranteed message

The entry “guaranteed message” in the table of return codes indicates the following:

The message code and the inserts (numbering and assignment of contents) are guaranteed to appear for certain messages.

If guaranteed messages exist for a particular return code, the number of the message will follow the text which briefly describes the code.

Commands

ASSIGN-STREAM
Assign S variable stream

The ASSIGN-STREAM command is used to assign an S variable stream for structured outputs to an (output) server that controls further processing of the variable stream.

```
ASSIGN-STREAM

STREAM-NAME = SYSVAR / SYSMSG / SYSINF / <structured-name 1..20>
,TO = *STD / <structured-name 1..20> / *DUMMY / *SAME-AS-CALLING-PROC / *VARIABLE(...) / *SERVER(...)
   *VARIABLE(...)
      VARIABLE-NAME = *NONE / <composed-name 1..255>(...)
         <composed-name 1..255>(...)
            WRITE-MODE = *EXTEND / *PREFIX
      ,RETURN-VARIABLE-NAME = *NONE / <composed-name 1..255>(...)
         <composed-name 1..255>(...)
            WRITE-MODE = *EXTEND / *PREFIX
      ,CONTROL-VAR-NAME = *NONE / <composed-name 1..255>(...)
         <composed-name 1..255>(...)
            WRITE-MODE = *EXTEND / *PREFIX
      ,RET-CONTROL-VAR-NAME = *NONE / <composed-name 1..255>(...)
         <composed-name 1..255>(...)
            WRITE-MODE = *EXTEND / *PREFIX
   *SERVER(...)
      SERVER-NAME = <structured-name 1..30>
      ,SERVER-INFORMATION = *NONE / <c-string 1..1800>
```

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	SDP0531	Warning returned by server; process continuing
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	64	SDP0532	Server error; command rejected
	64	SDP0534	Internal server error; command terminated. Server link terminated following unexpected event or due to shortage or absence of system resources
	130	SDP0099	No further address space available

BEGIN-BLOCK
Initiate command block

Command blocks which are to be treated as a logical unit begin with the BEGIN-BLOCK command and end with the END-BLOCK command. These command blocks are also called BEGIN blocks.

BEGININ-**BLOCK**

PROGRAM-**INP**UT = ***STD** / ***MIX**ED-**WITH-CMD**(...)
  ***MIX**ED-**WITH-CMD**(...)
  ⎸ **PROPA**GATE-**STMT-RC** = ***STD** / ***TO-CMD-RC**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

BEGIN-PARAMETER-DECLARATION
Declare procedure parameters

The procedure parameters are declared with the DECLARE-PARAMETER command in the procedure head. If the DECLARE-PARAMETER command is to be called several times, these calls are enclosed in a command block which begins with the BEGIN-PARAMETER-DECLARATION command and ends with the END-PARAMETER-DECLARATION command.

BEGIN-**PAR**AMETER-**DECL**ARATION

Return codes

Return codes will be supplied by this command only when it is used outside the procedure head. Errors in the procedure head are recognized by SDF-P during pre-analysis and result in the procedure run being aborted. .

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

BEGIN-STRUCTURE
Declare static structure

If a structure layout is declared, BEGIN-STRUCTURE identifies the beginning of the structure layout declaration. The structure layout must be declared before the static structures which are to correspond to it. The declaration of the structure layout is terminated with the END-STRUCTURE command.

If a static structure is declared with *BY-SYSCMD, the BEGIN-STRUCTURE command must directly follow the DECLARE-VARIABLE command in which the structure is declared. In this case, the command initiates the element declarations.

BEGIN-STRUCTURE
NAME = ***NONE** / <structured-name 1..20>(...)
<structured-name 1..20>(...)
⏐ **SCOPE** = ***CURRENT** / ***PROCEDURE** / ***TASK**(...)
⏐   ***TASK**(...)
⏐   ⏐ **STATE** = ***ANY** / ***NEW**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

CALL-PROCEDURE
Start command sequence

The CALL-PROCEDURE command starts a stored command sequence (procedure). During processing, symbolic parameters contained in the sequence are replaced by the values specified in the command call (PROCEDURE-PARAMETERS operand).

Note

This command is a component part of BS2000/OSD-BC (status of description: V7.0). In contrast to the SDF-P commands, the caller requires the STD-PROCESSING or HARDWARE-MAINTENANCE privilege.

CALL-PROCEDURE	Alias: **CL** / **CLP**
FROM-FILE = <filename 1..54 without-gen> / ***LIB**RARY-**ELEM**ENT(...) / ***VAR**IABLE(...)	
***LIB**RARY-**ELEM**ENT(...)	
LIBRARY = <filename 1..54 without-gen>	
,**ELEM**ENT = <composed-name 1..64>(...)	
<composed-name 1..64>(...)	
VERSION = ***HIGH**EST-**EXIST**ING / <composed-name 1..24>	
,**TYPE** = ***STD** / ***BY-LATEST-MODIFICATION** / <alphanum-name 1..8>	
***VAR**IABLE(...)	
VARIABLE-**NAME** = <composed-name 1..255>	
,**PROC**EDURE-**PAR**AMETERS = ***NO** / <text 0..1800 with-low>	
,**LOG**GING = ***PAR**AMETERS(...) / **Y**ES / ***NO** /	
***PAR**AMETERS(...)	
CMD = ***BY-PROC-TEST-OPT**ION / ***Y**ES / ***NO**	
,**DATA** = ***BY-PROC-TEST-OPT**ION / ***Y**ES / ***NO**	
,**UNL**OAD-**ALLOW**ED = ***Y**ES / ***NO**	
,**EXEC**UTION = ***Y**ES / ***NO**	

Return codes

The following command return codes can only be returned if the called procedure does not supply any command return code itself (e.g. EXIT-PROCEDURE not executed due to an error). Command return codes whose maincode begins with "SSM" can only be returned when a non-S procedure is called.
Command return codes whose maincode begins with "SDP" can only be returned when an S procedure is called.

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	SSM2058	Protocol type error
2	0	SSM2065	EOF on procedure file, /END-PROC simulated
	1	SSM2036	Incomplete operand
	1	SSM2054	Symbolic operand error
	1	SSM2055	Symbolic operand error in /BEGIN-PROC
	1	SDP0138	Error in pre-analysis of text procedure, or object procedure invalid
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0093	Non-S procedure can only be type J element
	64	SDP0144	Error on parameter transfer
	64	SSM2052	DMS error (Open error)
	64	SSM2053	Not a SAM/ISAM file or file does not begin with /BEGIN-PROC or /PROC
	64	SSM2056	/CALL-PROC and /BEGIN-PROC parameters incompatible
	64	SSM2061	Error on accessing library element
	64	SSM2064	Procedure file cannot be fetched by remote processor
	130	SDP0099	No further address space available
xx	xx	xxxxxxx	Other return codes from the called procedure

CLOSE-VARIABLE-CONTAINER
Close variable container

The CLOSE-VARIABLE-CONTAINER command closes the specified variable containers.

CLOSE-VARIABLE**-CONTAINER**
CONTAINER-NAME = <composed-name 1..64 with-wild(80)> / list-poss(2000): <composed-name 1..64>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax errorr
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

COMPILE-PROCEDURE
Compile procedure

The COMPILE-PROCEDURE command converts an S procedure into a compiled procedure, i.e. into an intermediate format that can be used in environments in which the SDF-P subsystem is not available.

The full functional scope of SDF-P can be used in compiled procedures. This also holds true if these procedures are started in an environment containing only SDF-P-BASYS.

```
COMPILE-PROCEDURE

FROM-FILE = <filename 1..54 without-gen> / *LIBRARY-ELEMENT(...)
  *LIBRARY-ELEMENT(...)
    | LIBRARY = <filename 1..54 without-gen>
    | ,ELEMENT = <composed-name 1..64>(...)
    |    <composed-name 1..64>(...)
    |       | VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24>
    | ,TYPE = J / <alphanum-name 1..8>
,TO-FILE = <filename1..54without-gen> / *LIBRARY-ELEMENT(...) / *DUMMY
  *LIBRARY-ELEMENT(...)
    | LIBRARY = *SAME / <filename 1..54 without-gen>
    | ,ELEMENT = *SAME(...) / <composed-name 1..64>(...)
    |    *SAME(...)
    |       | VERSION = *SAME / *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
    |       |                     <composed-name 1..24>
    |    <composed-name 1..64>(...)
    |       | VERSION = *SAME / *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
    |       |                     <composed-name 1..24>
    | ,TYPE = SYSJ / <alphanum-name 1..8>
```

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0138	Error during procedure preanalysis Guaranteed message: SDP0138
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

CYCLE
Terminate loop pass

The CYCLE command can be called in loop blocks (FOR, WHILE, REPEAT block). It then terminates the current loop pass and resumes procedure execution by executing the terminating command in the loop block (END-FOR, END-WHILE, UNTIL). The loop condition is then rechecked and, if necessary, the next loop pass started .
Execution of the CYCLE command can be made subject to a condition.

CYCLE
BLOCK = ***LAST** / ***ALL** / <structured-name 1..255> ,**COND**ITION = ***NONE** / <text 1..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error (incorrect expression)
	130	SDP0099	No further address space available

DECLARE-CONSTANT
Declare variable with constant value

The DECLARE-CONSTANT command is used to declare one or more variables and to assign them a constant value, thus protecting these values from being overwritten. Variables with a constant value are treated in much the same way as ordinary variables. You cannot, however, modify their value using SET-VARIABLE or remove the value using FREE-VARIABLE.

DECLARE-CONSTANT
VARIABLE-**NAME** = list-poss(2000): <structured-name 1..20>(...)
<structured-name 1..20>(...)
| **VAL**UE = <text 0..1800 with-low *expr*>
| ,**TYPE** = ***ANY*** / ***STRING** / ***INTEGER** / ***BOOL**EAN
,**SCOPE** = ***CURR**ENT(...) / ***PROC**EDURE(...) / ***TASK**(...)
***CURR**ENT(...)
| **IMPORT-ALLOWED** = ***NO*** / ***YES**
***PROC**EDURE(...)
| **IMPORT-ALLOWED** = ***NO*** / ***YES**
***TASK**(...)
| **STATE** = ***ANY*** / ***NEW** / ***OLD**
,**CONTAINER** = ***STD*** / <composed-name 1..64> **/ *VAR**IABLE(...)
***VAR**IABLE(...)
| **VAR**IABLE-**NAME** = <structured-name 1..20>
| ,**SCOPE** = ***VISIBLE*** / ***TASK**

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Warning; element already declared
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error Guaranteed messages: SDP1018, SDP1030
	130	SDP0099	No further address space available

DECLARE-ELEMENT
Declare structure element

Structure elements can be simple or complex variables (arrays, structures, lists).
Variable attributes which cannot be defined in the DECLARE-ELEMENT command are taken from the superordinate structure (e.g. the SCOPE attribute of BEGIN-STRUCTURE or DECLARE-VARIABLE).

If the structure element is a complex variable, its elements must be initialized individually. Complex variables cannot be initialized in their entirety. This command can also be used to declare elements of dynamic structures.

```
DECLARE-ELEMENT

NAME = list-poss(2000): <composed-name 1..255>(...)
  <composed-name 1..255>(...)
  | INITIAL-VALUE = *NONE / <text 0..1800 with-low expr>
  | ,TYPE = *ANY / *STRING / *INTEGER / *BOOLEAN / *STRUCTURE(...)
  |    *STRUCTURE(...)
  |      | DEFINITION = *DYNAMIC / *BY-SYSCMD / <structured-name 1..20>
,MULTIPLE-ELEMENTS = *NO / *ARRAY(...) / *LIST(...)
  *ARRAY(...)
  | LOWER-BOUND = 0 / *NONE / <integer -2147483648..2147483647>
  | ,UPPER-BOUND = *NONE / <integer -2147483648..2147483647>
  *LIST(...)
  | LIMIT = *NONE / <integer 1..2147483647>
```

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Warning; element already declared
2	0	CMD0001	Warning; INITIAL-VALUE operand was ignored
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error Guaranteed messages: SDP1018
	130	SDP0099	No further address space available

DECLARE-PARAMETER
Declare procedure parameters

The procedure parameters needing an actual value during the procedure run are declared with the DECLARE-PARAMETER command. Furthermore, the way the parameter values are passed to the procedure is defined (initial value, prompting, ...). Procedure parameters may only be declared in the procedure header.

Procedure parameters are variables local to the procedure in SDF-P: When defined in the procedure header, they implicitly have SCOPE = *CURRENT.
The names of the procedure parameters are also keywords for the procedure parameters in the PROCEDURE-PARAMETERS operand of the CALL-PROCEDURE, ENTER-PROCEDURE and INCLUDE-PROCEDURE commands.

```
DECLARE-PARAMETER

NAME = list-poss(2000): <structured-name 1..20>(...)

  <structured-name 1..20>(...)
   |   INITIAL-VALUE = *NONE / *PROMPT(...) / <text 0..1800 with-low expr>
   |     *PROMPT(...)
   |       |  PROMPT-STRING = *STD / <text 0..1800 with-low string-expr>
   |       | ,DEFAULT-VALUE = *NONE / <text 0..1800 with-low expr>
   |       | ,SECRET-INPUT  = *NO / *YES
   |  ,TYPE = *ANY / *STRING / *INTEGER / *BOOLEAN
   |  ,TRANSFER-TYPE = *BY-VALUE / *BY-REFERENCE
```

Return codes

The following return codes can thus appear only if DECLARE-PARAMETER is used in another (i.e. wrong) context:

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

DECLARE-VARIABLE
Declare variable

DECLARE-VARIABLE is used to define the attributes of this variable and possibly an initial value as well.
Job variables can be integrated in SDF-P via the CONTAINER operand.

```
DECLARE-VARIABLE                                                    Alias: DCV

VARIABLE-NAME = list-poss(2000): <structured-name 1..20>(...)
  <structured-name 1..20>(...)
    | INITIAL-VALUE = *NONE / <text 0..1800 with-low expr>
    | ,TYPE = *ANY / *STRING / *INTEGER / *BOOLEAN / *STRUCTURE(...)
    |   *STRUCTURE(...)
    |     | DEFINITION = *DYNAMIC / *BY-SYSCMD / <structured-name 1..20>
,MULTIPLE-ELEMENTS = *NO / *ARRAY(...) / *LIST(...)
  *ARRAY(...)
    | LOWER-BOUND = 0 / *NONE / <integer -2147483648..2147483647>
    | ,UPPER-BOUND = *NONE / <integer -2147483648..2147483647>
  *LIST(...)
    | LIMIT = *NONE / <integer 1..2147483647>
,SCOPE = *CURRENT(...) / *PROCEDURE(...) / *TASK(...)
  *CURRENT(...)
    | IMPORT-ALLOWED = *NO / *YES
  *PROCEDURE(...)
    | IMPORT-ALLOWED = *NO / *YES
  *TASK(...)
    | STATE = *ANY / *NEW / *OLD
,CONTAINER = *STD / <composed-name 1..64> / *VARIABLE(...) / *JV(...)
  *VARIABLE(...)
    | VARIABLE-NAME = <structured-name 1..20>
    | ,SCOPE = *VISIBLE / *TASK
  *JV(...)
    | JV-NAME = <filename 1..54>
    | ,STATE = *ANY / *NEW / *OLD
```

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Nothing executed; element already declared
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error Guaranteed messages: SDP1018, SDP1030
	130	SDP0099	No further address space available

DELETE-STREAM
Delete S variable stream

The DELETE-STREAM command deletes S variable streams. Their assignments will no longer be displayed by SHOW-STREAM-ASSIGNMENT.
The deletion is restricted to those streams which were created at the highest procedure level (in dialog mode). Any subsequent transmissions via the deleted stream will be rejected.

DELETE**-STREAM**
STREAM-NAME = <composed-name 1..20 with-wild(40)> / list-poss(100): <structured-name 1..20>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	SDP0516	Specified variable stream name does not exist, or the search pattern was not found; process continues
1	0	SDP0518	No match for wildcard. process continues
2	0	SDP0535	Warning from the server during deletion of a specified S variable steam; process continues
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	64	SDP0515	Specified variable stream not created at the highest level
	64	SDP0536	Server error; deletion of the specified S variable stream has been rejected
	64	SDP0537	Internal server error; deletion of the specified S variable stream has been rejected. Server link terminated following unexpected event or due to system resource shortage or error

DELETE-VARIABLE
Delete variable

DELETE-VARIABLE deletes the declaration of an S variable within the current scope, i.e. including the declarations of imported task variables.
Either simple or complex variables can be deleted, but not individual elements of complex variables.

DELETE-**VAR**IABLE
VARIABLE-**NAME** = <structured-name 1..20 with-wild(40)> / list-poss(2000): <structured-name 1..20>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
1	0	CMD0001	Warning; nothing executed
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

ELSE
Initiate ELSE branch in IF block

In the IF block, the ELSE command initiates the last branch. The commands between the ELSE command and the terminating END-IF command are executed if none of the conditions previously checked in the IF or ELSE-IF commands applies.
In the IF-BLOCK-ERROR and IF-CMD-ERROR blocks, the ELSE branch is executed if no errors occur

ELSE

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

ELSE-IF
Initiate alternative branch in IF block

The ELSE-IF branch is executed if the condition specified in the ELSE-IF command applies; otherwise the system searches for the next branch in the IF block or an END-IF command. The ELSE-IF branch contains all commands positioned between the current ELSE-IF command and the next ELSE-IF, ELSE or END-IF command.

ELSE-IF
CONDITION = <text 0..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

END-BLOCK
Terminate command block

END-BLOCK terminates a BEGIN block, i.e. a command block which was initiated with the BEGIN-BLOCK command.

END-BLOCK
BLOCK = ***LAST** / <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

END-FOR
Terminate FOR block

END-FOR terminates a FOR block, i.e. a FOR loop which was initiated with the FOR command.

END-FOR
BLOCK = ***LAST** / <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0139	Back branch limit reached
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

END-IF
Terminate IF block

END-IF terminates blocks with conditional command sequences, i.e.:
- IF block
- IF-BLOCK-ERROR block
- IF-CMD-ERROR block

END-IF
BLOCK = ***LAST** / <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

END-PARAMETER-DECLARATION
Terminate procedure parameter declaration

END-PARAMETER-DECLARATION terminates the command block which was initiated with the BEGIN-PARAMETER-DECLARATION command; the procedure parameters are declared in this block.

END-PARAMETER-**DECL**ARATION

Return codes

The following return codes can thus appear only if END-PARAMETER-DECLARATION is used in another (i.e. wrong) context:

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

END-STRUCTURE
Identify end of structure declaration

END-STRUCTURE terminates a structure declaration block which was initiated with BEGIN-STRUCTURE.

END-STRUCTURE
NAME = ***LAST** / <structured-name 1..20>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	CMD0001	Warning; structure is empty
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

END-WHILE
Terminate WHILE block

END-WHILE terminates a WHILE block, i.e. a loop which was initiated with the WHILE command.
The loop condition in the WHILE command is checked during execution of the END-WHILE command. If the condition is met (TRUE), the first command in the WHILE block is used to start the next loop pass. Otherwise, the loop is terminated. Procedure execution resumes with the first command following END-WHILE.

END-WHILE
BLOCK = ***LAST** / <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0139	Back branch limit reached
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

ENTER-PROCEDURE
Start procedure in background as batch job

The ENTER-PROCEDURE command can be used to start a procedure as a batch job. The command is part of the BS2000/OSD configuration (as of BS2000/OSD V7.0).

(part 1 of 2)

ENTER-PROCEDURE Alias: **ENP**

FROM-FILE = <filename 1..54 without-gen> / ***LIB**RARY-**ELEM**ENT(...)

  ***LIB**RARY-**ELEM**ENT(...)

    LIBRARY = <filename 1..51 without-gen>
    ,**ELEM**ENT = <composed-name 1..38>

,**PROC**EDURE-**PAR**AMETERS = ***NO** / <text 0..1800 with-low>

,**PROC**ESSING-**ADMISS**ION = ***SAME** / ***PAR**AMETERS(...)

  ***PAR**AMETERS(...)

    USER-IDENTIFICATION = ***NONE** / <name 1..8>
    ,**ACCOUNT** = ***NONE** / <alphanum-name 1..8>
    ,**PASS**WORD = ***NONE** / <c-string 1..8> / <c-string 9..32> / <x-string 1..16> / ***SECRET**

,**PROC**EDURE-**PASS**WORD = ***NONE** / <x-string 1..8> / <c-string 1..4> /
      <integer -2147483648..2147483647> / ***SECRET**

,**CRYPTO-PASS**WORD = ***NONE** / <c-string 1..8> / <x-string 1..16> / ***SECRET**

,**HOST** = ***STD** / <c-string 1..8> / ***ANY**

,**JOB-CLASS** = ***STD** / <name 1..8>

,**JOB-NAME** = ***NO** / <name 1..8>

,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>

,**JV-PASS**WORD = ***NONE** / <c-string 1..4> / <x-string 1..8> / ***SECRET** /
      <integer -2147483648..2147483647>

,**JOB-PRIO**RITY = ***STD** / <integer 1..9>

,**RERUN-AF**TER-**CRASH** = ***NO** / ***Y**ES

,**FLUSH-AF**TER-**SHUTDOWN** = ***NO** / ***Y**ES

,**SCHEDUL**ING-**TIME** = ***STD** / ***PAR**AMETERS(...) / ***BY-CALENDAR**(...)

  ***PAR**AMETERS(...)

    START = ***STD** / ***SOON** / ***IMMEDI**ATELY / ***AT-STREAM**-STARTUP / ***WITHIN**(...) / ***AT**(...) /
      ***EARL**IEST(...) / ***LATEST**(...)

      ***WITHIN**(...)

        HOURS = **0** / <integer 0..23 *hours*>
        ,**MINUT**ES = **0** / <integer 0..59 *minutes*>

      ***AT**(...)

        DATE = ***TODAY** / <date>
        ,**TIME** = <time>

continued ➠

(part 2 of 2)

***EARL**IEST(...)

DATE = ***TODAY** / <date>

,**TIME** = <time>

***LATEST**(...)

DATE = ***TODAY** / <date>

,**TIME** = <time>

,**REP**EAT**-JOB** = ***STD** / ***NO** / ***DAILY** / ***WEEKLY** / ***AT-STREAM**-STARTUP / ***PER**IOD(...)

***PER**IOD(...)

HOURS = **0** / <integer 0..23 *hours*>

,**MINUT**ES = **0** / <integer 0..59 *minutes*>

***BY-CALENDAR**(...)

CALENDAR-NAME = <filename 1..54 without-gen-vers>

,**SYMB**OLIC**-DATE** = <filename 1..20 without-cat-user-vers> / <partial-filename 2..20 without-cat-user>

,**LIMIT** = ***STD** / <integer 1..32767> / ***BY-DATE**(...)

***BY-DATE**(...)

DATE = <date>

,**TIME** = <time>

,**RESOURCES** = ***PAR**AMETERS (...)

***PAR**AMETERS(...)

RUN-PRIORITY = ***STD** / <integer 30..255>

,**CPU-LIM**IT = ***STD** / ***NO** / <integer 1..32767 *seconds*>

,**SYSLST-LIM**IT = ***STD** / ***NO** / <integer 0..999999>

,**SYSOPT-LIM**IT = ***STD** / ***NO** / <integer 0..999999>

,**LOG**GING = ***STD** / ***Y**ES / ***NO**

,**LIST**ING = ***NO** / ***Y**ES

,**JOB-PAR**AMETER = ***NO** / <c-string 1..127>

,**SYS**TEM**-OUT**PUT = ***STD** / ***PRINT** / ***DEL**ETE

,**ASS**IGN**-SYS**TEM**-FILE**S = ***STD** / ***PAR**AMETERS(...)

***PAR**AMETERS(...)

SYSLST = ***STD** / ***PRIM**ARY / ***DUMMY** / <filename 1..54>

,**SYSOUT** = ***STD** / ***PRIM**ARY / ***DUMMY** / <filename 1..54>

,**PROT**ECTION = ***NONE** / ***CANCEL**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	Command executed
2	0	CMD0002	Command executed with warning e.g. DELETE=*YES ignored for repeat job or when a library element is specified
	1	CMD0202	Syntax error in command
	32	CMD0221	System error
	64	JMS0630	Semantic or privileges error e.g. processor, catalog ID, job class unknown; MONJV not accessible
	64	JMS0640	Error on accessing the procedure file e.g. not a SAM or ISAM file, file empty, missing access right
	64	JMS0670	Error in a remote job
	130	JMS0620	No further storage space or TSN available; or specified MONJV is already monitoring a job
	130	JMS0650	MSCF not available, or no connection with the specified processor

EXECUTE-CMD
Execute command and structured output

The EXECUTE-CMD command passes the command specified with the operand CMD=... on for execution and writes the command output (messages, output information) to a specified variable. Both structured and unstructured output can be generated, depending on the capabilities of the command server. The reaction to errors occurring in command execution can be controlled by evaluation of the return code.

EXECUTE-**CMD**
CMD = <text 0..1800 with-low>
,**TEXT-OUTPUT** = ***SYSOUT** / ***NONE** / <composed-name 1..255>(...)
<composed-name 1..255>(...)
| **WRITE-MODE** = ***REPLACE** / ***EXTEND**
,**STRUCTURE-OUTPUT** = ***NONE** / <composed-name 1..255>(...)
<composed-name 1..255>(...)
| **WRITE-MODE** = ***REPLACE** / ***EXTEND**
,**MSG-STRUCTURE-OUTPUT** = ***NONE** / <composed-name 1..255>(...)
<composed-name 1..255>(...)
| **WRITE-MODE** = ***REPLACE** / ***EXTEND** /
,**RET**URNCODE = ***STD** / ***NONE** / ***VAR**IABLE(...)
***VAR**IABLE(...)
| **SUBCODE1** = ***NONE** / <composed-name 1..255>
| ,**SUBCODE2** = ***NONE** / <composed-name 1..255>
| ,**MAIN**CODE = ***NONE** / <composed-name 1..255>

Return codes

*The following return codes are possible if RETURNCODE = *STD*:

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available
xx	xx	xxxxxxx	other return codes from the executed commands

*The following return codes are possible if RETURNCODE = *NONE / *VARIABLE(...):*

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error (but only in EXECUTE-CMD)
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

EXIT-BLOCK
Terminate processing of command block

EXIT-BLOCK terminates processing of a command block (BEGIN, IF, WHILE, REPEAT block, etc.) and resumes procedure execution at the command following the block termination command. For a BEGIN-Block that was called with a INCLUDE-BLOCK command, execution of the procedure continues at the command that follows the INCLUDE-BLOCK command. Execution of the EXIT-BLOCK command can be made to depend on a condition.

EXIT-BLOCK
BLOCK = **<u>*LAST</u>** / ***ALL** / <structured-name 1..255>
,**COND**ITION = **<u>*NONE</u>** / <text 1..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

EXIT-PROCEDURE
Terminate procedure

EXIT-PROCEDURE terminates the current procedure and returns job control to the calling level.

The specification ERROR=*YES initiates error handling if SUBCODE1 is not zero.

EXIT-PROCEDURE
ERROR = ***NO** (...) / ***Y**ES(...)
***NO**(...)
SUBCODE2 = **0** / < integer 0..255>
,**MAIN**CODE = **CMD0001** / <alphanum-name 7..7>
***Y**ES(...)
SUBCODE1 = **64** / <integer 0..255>
,**SUBCODE2** = **0** / <integer 0..255>
,**MAIN**CODE = **SDP0018** / <alphanum-name 7..7>
,**RESU**ME-**PROG**RAM = ***NO** / ***Y**ES

Return codes

If execution of the EXIT-PROCEDURE command itself results in an error, control is not returned to the caller; instead, one of the following return codes is passed:

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

FOR
Initiate FOR block

The FOR command initiates a FOR block, which subsequently ends with the /END-FOR command. The entire construct is a FOR loop.

FOR
<composed-name 1..255> = list-poss(2000): ***LIST**(...) / ***COUNTER** (...) / <text 0..1800 with-low *expr*>
***LIST**(...)
│ **LIST-NAME** = <composed-name 1..255>
***COUNTER**(...)
│ **FROM** = <text 0..1800 *arithm-expr*>
│ ,**TO** = ***UNLIMITED** / <text 0..1800 *arithm-expr*>
│ ,**INCREMENT** = **1** / <text 0..1800 *arithm-expr*>
,**CONDITION** = ***NONE** / <text 0..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

FREE-VARIABLE
Delete contents of variable

The FREE-VARIABLE command deletes the contents of one or more S variables.

The FREE-VARIABLE command can be used on simple and composed variables. One or more elements can be deleted from list variables with this command. For variables with a dynamic structure, not only the contents are deleted, but also the element itself.

FREE-VARIABLE
VARIABLE-**NAME** = <structured-name 1..20 with-wild(40)> / ***LIST**(...) / list-poss(2000): <composed-name 1..255> ***LIST**(...) **LIST-NAME** = <composed-name 1..255> ,**FROM-INDEX** = ***FIRST** / ***LAST** / <integer 1..214783647> ,**NUMBER-OF-ELEM**ENTS = **1** / ***REST** / <integer 1..214783647> ,**DESTROY-CONTAINER**-JV = ***NO** / ***Y**ES

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Warning: no elements deleted; variable has already been deleted
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error Guaranteed messages: SDP1008
	130	SDP0099	No further address space available

GOTO
Branch to tag

GOTO branches to the specified tag and continues procedure execution that location. The tag must be defined in the same or a surrounding block (see Example 2). In addition, the tag must be unique to the procedure.

GOTO
LABEL = <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0139	Back branch limit reached
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

IF
Initiate IF block

The IF command initiates an IF block, i.e. a conditional command sequence. The IF block must be terminated by the END-IF command.

IF
CONDITION = <text 0..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

IF-BLOCK-ERROR
Initiate block error handling

IF-BLOCK-ERROR initiates block error handling.
An ELSE branch can be defined in the IF-BLOCK-ERROR block, using the ELSE command.
The IF-BLOCK-ERROR block is terminated with the END-IF command.

IF-BLOCK-ERROR

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

IF-CMD-ERROR
Initiate command error handling

IF-CMD-ERROR initiates a command sequence which is executed when an error occurs in the directly preceding command. This permits specific error handling for this command, thus avoiding block error handling. The IF-CMD-ERROR block must be terminated by the END-IF command

IF-CMD-ERROR

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

IMPORT-VARIABLE
Import variable

The IMPORT-VARIABLE command is used to import a previously declared variable into the called procedure. It is equivalent in many ways to DECLARE-VARIABLE, but eliminates the need to repeatedly list all assigned attributes.

IMPORT**-VAR**IABLE

VARIABLE**-NAME** = <structured-name 1..20 with-wild(40)> / list-poss(2000): <structured-name 1..20>
,**FROM** = ***SCOPE**(...)
  ***SCOPE**(...)
    │ **SCOPE** = ***TASK** / ***CALL**ING**-PROC**EDURES

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Warning: element already declared
2	0	SDP2000	Warning: not all elements of the input list could be processed successfully. Guaranteed message: SDP2000
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error Guaranteed messages: SDP1008, SDP1018
	64	SDP2001	None of the elements could be imported
	130	SDP0099	No further address space available

INCLUDE-BLOCK
Executing a BEGIN block as a subprocedure

The INCLUDE-BLOCK command jumps to the BEGIN-BLOCK command with the specified tag (name). After executing the (next) END-BLOCK or EXIT-BLOCK command, execution returns to the command following the INCLUDE-BLOCK command.
The INCLUDE-BLOCK command allows you to comfortably execute a subprocedure defined between the BEGIN-BLOCK and END-BLOCK commands.

The INCLUDE-BLOCK command is rejected in the dialog mode.

INCLUDE-BLOCK
BLOCK = <structured-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)

INCLUDE-CMD
Call command sequence from program

The INCLUDE-CMD command is used to include a command or command sequence from a program for execution. The command can be executed only in the CMD macro (TU program) and in EXECUTE-SYSTEM-CMD statements.

INCLUDE-CMD	Alias: **INCMD**
CMD = <text 0..1800 with-low>	

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0138	Error during procedure preanalysis
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

INCLUDE-PROCEDURE
Start command sequence as include procedure

The INCLUDE-PROCEDURE command starts a stored command sequence (procedure). An include procedure is a procedure which is visible in the data environment of the calling procedure. During processing, symbolic parameters contained in the sequence are replaced by the values specified in the command call (PROCEDURE-PARAMETERS operand).

```
INCLUDE-PROCEDURE                                                          Alias: INP

FROM-FILE = <filename 1..54 without-gen> / *LIBRARY-ELEMENT(...) / *VARIABLE(...)
  *LIBRARY-ELEMENT(...)
    | LIBRARY = <filename 1..54 without-gen>
    | ,ELEMENT = <composed-name 1..64>(...)
    |    <composed-name 1..64>(...)
    |      | VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
    | ,TYPE = *STD / *BY-LATEST-MODIFICATION / <alphanum-name 1..8>
  *VARIABLE(...)
    | VARIABLE-NAME = <composed-name 1..255>
,PROCEDURE-PARAMETERS = *NO / <text 0..1800 with-low expr>
,LOGGING = *PARAMETERS(...) / YES / *NO /
  *PARAMETERS(...)
    | CMD = *BY-PROC-TEST-OPTION / *YES / *NO
    | ,DATA = *BY-PROC-TEST-OPTION / *YES / *NO
,UNLOAD-ALLOWED = *YES / *NO
,EXECUTION = *YES / *NO
```

Return codes

The following command return codes can only be returned if the called procedure does not supply any command return code itself (e.g. EXIT-PROCEDURE not executed due to an error).
Command return codes whose maincode begins with "SSM" can only be returned when a non-S procedure is called.
Command return codes whose maincode begins with "SDP" can only be returned when an S procedure is called.

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
2	0	SSM2058	Protocol type error
2	0	SSM2065	EOF on procedure file, /END-PROC simulated
	1	SSM2036	Incomplete operand
	1	SSM2054	Symbolic operand error
	1	SSM2055	Symbolic operand error in /BEGIN-PROC

(SC2)	SC1	Maincode	Meaning
	1	SDP0138	Error in pre-analysis of text procedure, or object procedure invalid
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0093	Non-S procedure can only be type J element
	64	SDP0144	Error on parameter transfer
	64	SSM2052	DMS error (Open error)
	64	SSM2053	Not a SAM/ISAM file or file does not begin with /BEGIN-PROC or /PROC
	64	SSM2056	Parameters of /INCL-PROC and /BEGIN-PROC do not mach
	64	SSM2061	Error on accessing library element
	64	SSM2064	Procedure file cannot be fetched by remote processor
	130	SDP0099	No further address space available
xx	xx	xxxxxxx	Other return codes from the called procedure

MODIFY-PROCEDURE-OPTIONS
Modify procedure attributes during procedure execution

MODIFY-PROCEDURE-OPTIONS can be used to modify, during procedure execution, most of the procedure attributes set with SET-PROCEDURE-OPTIONS at the beginning of the procedure execution.

```
MODIFY-PROCEDURE-OPTIONS

 IMPLICIT-DECLARATION = *UNCHANGED / *YES / *NO
,LOGGING-ALLOWED = *PARAMETERS(...) / *NO / *YES
   *PARAMETERS(...)
     |  CMD = *UNCHANGED / *YES / *NO
     | ,DATA = *UNCHANGED / *YES / *NO
,INTERRUPT-ALLOWED = *UNCHANGED / *YES / *NO
,DATA-ESCAPE-CHAR = *UNCHANGED / *NONE / '&&' / '#' / '*' / '@' / '$' / *STD
,DATA-ERROR-HANDLING = *UNCHANGED / *YES / *NO
,JV-REPLACEMENT = *UNCHANGED / *NONE / *AFTER-BUILTIN-FUNCTION
,ERROR-MECHANISM = *UNCHANGED / *SPIN-OFF-COMPATIBLE / *BY-RETURNCODE
,SUPPRESS-SDP-MSG = *UNCHANGED / *NONE / *ADD(...) / *REMOVE(...)
   *ADD(...)
     | MSG-ID = list-poss(2000): <alphanum-name 7..7>)
   *REMOVE(...)
     | MSG-ID = list-poss(2000): <alphanum-name 7..7>)
```

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

MODIFY-PROCEDURE-TEST-OPTIONS
Modify logging and limit number of back branches

The following settings for executing the procedure can be changed for test purposes with the MODIFY-PROCEDURE-TEST-OPTIONS command:

- Logging of commands and data
- Limiting back branches prevents
- Simulation of SDF-P-BASYS

MODIFY-**PROC**EDURE-**TEST-OPT**IONS
LOGGING = ***PAR**AMETERS(...) / ***Y**ES / ***NO** /
***PAR**AMETERS(...)
CMD = ***UNCHA**NGED / ***Y**ES / ***NO**
,**DATA** = ***UNCHA**NGED / ***Y**ES / ***NO**
,**BACK-BRANCH-LIMIT** = ***UNCHA**NGED / ***NONE** / <text 0..1800 with-low *arith-expr*>
,**FUNCTIONALITY** = ***UNCHA**NGED / ***FULL** / ***BASIC**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	64	SDP0091	System error (internal error)
	64	SDP0133	Semantic error
	32	CMD0221	Invalid data type in expression
	130	SDP0099	No further address space available

OPEN-VARIABLE-CONTAINER
Open variable container

The OPEN-VARIABLE-CONTAINER command is used to open variable containers, which are stored as PLAM library elements. If such a variable container or element does not yet exist when the command is called, it is automatically created.

```
OPEN-VARIABLE-CONTAINER

 CONTAINER-NAME = <composed-name 1..64>
,FROM-FILE = *LIBRARY-ELEMENT (...)
   *LIBRARY-ELEMENT(...)
   |  LIBRARY = <filename 1..54 without-vers>
   | ,ELEMENT = *CONTAINER-NAME / <composed-name 1..64>(...)
   |     <composed-name 1..64>(...)
   |     |  VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
,LOCK-ELEMENT = *NO / *YES
,SCOPE = *CURRENT / *PROCEDURE / *TASK(...)
   *TASK(...)
   |  SAVE-AT-TERMINATION = *NO / *YES
,AUTOMATIC-DECLARE = *ALL / *NONE / <structured-name 1..20 with-wild(40)> /
                     list-poss(2000): <structured-name 1..20>
```

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
2	0	SDP00xx	Warning that the following has occurred: guaranteed messages: SDP1008, SDP1018
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

RAISE-ERROR
Generate return code

RAISE-ERROR generates a command return code and subsequently activates error handling if SUBCODE 1 is a number other than zero.

RAISE-ERROR
SUBCODE1 = **<u>64</u>** / <integer 0..255> ,**SUBCODE2** = **<u>0</u>** / <integer 0..255> ,**MAIN**CODE = **<u>SDP0018</u>** / <alphanum-name 7..7>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available
xx	xx	xxxxxxx	Return code as specified in operands

READ-VARIABLE
Assign values to variables

The READ-VARIABLE command is used to read data from an input medium and to store the data in a variable. The data is read record by record. The operation is terminated when the string *END-OF-CMD or the end of a file (EOF) is encountered.

```
READ-VARIABLE                                                          Alias: RDV

VARIABLE-NAME = *BY-INPUT(...) / *LIST(...) / list-poss(2000): <composed-name 1..255>
  *BY-INPUT(...)
    | PREFIX = *NONE / <composed-name 1..255>
  *LIST(...)
    | LIST-NAME = <composed-name 1..255>
    | ,WRITE-MODE = *REPLACE / *EXTEND
,STRING-QUOTES = *OPTIONAL / *YES / *NO
,INPUT = *TERMINAL(...) / <filename 1..54 without-gen-vers>(...) / *VARIABLE(...) /
         *LIBRARY-ELEMENT(...) / *SYSDTA(...)
  *TERMINAL(...)
    | PROMPT-STRING = '>>' / <text 0..1800 with-low>
    | ,SECRET-INPUT = *NO / *YES
  <filename 1..54 without-gen-vers>(...)
    | REMOVE-KEY = *YES / *NO
    | ,BEGIN-RECORD = *FIRST / <integer 1..2147483647>
    | ,END-RECORD = *LAST / <integer 1..2147483647>
    | ,BEGIN-COLUMN = *FIRST / <integer 1..2147483647>
    | ,END-COLUMN = *LAST / <integer 1..2147483647>
    | ,PATTERN = *NONE / <c-string 0..1800 with-low>
    | ,PATTERN-TYPE = *STRING / *REGULAR-EXPRESSION
  *VARIABLE(...)
    | VARIABLE-NAME = <composed-name 1..255>
  *LIBRARY-ELEMENT(...)
    | LIBRARY = <filename 1..54 without-vers>
    | ,ELEMENT = <composed-name 1..64>(...)
    |    <composed-name 1..64>(...)
    |      | VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
    | ,TYPE = S / <alphanum-name 1..8>
  *SYSDTA(...)
    | REMOVE-KEY = *YES / *NO
```

Return codes

It is possible that part of the command has already been processed and executed before the error occurs. In this case, the result of the command is not guaranteed. .

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
2	0	SDP2000	Warning: not all elements of the input list could be processed successfully. Guaranteed message: SDP2000
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0089	INPUT error
	64	SDP0091	Semantic error Guaranteed messages: SDP1008
	64	SDP2001	None of the elements could be read in
	130	SDP0099	No further address space available

REPEAT
Initiate REPEAT block

REPEAT initiates a REPEAT block (a REPEAT loop). Execution of the command sequence within the REPEAT block is repeated until the loop condition is met.
The loop condition is checked in the UNTIL command which terminates the REPEAT block.

REPEAT

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

REPEAT-CMD
Repeat a command

The REPEAT-CMD command allows you to repeat the execution of a command. During the repetition of the command, special parts of the command (name part, operands, operand values) are substituted one after the other by the elements of an input list. These elements can be specified as records in a file, a library element or as list elements of an S variable, or they can be entered directly at the terminal. The command specified will be called for each element in the input list.

```
REPEAT-CMD                                                              Alias: REPCMD

CMD = <text 0..1800 with-low>
,SUBSTITUTION-LIST = *TERMINAL / <filename 1..54 without-gen-vers> / *VARIABLE(...) /
              *LIBRARY-ELEMENT(...)
    *VARIABLE(...)
    |   VARIABLE-NAME = <composed-name 1..255>
    *LIBRARY-ELEMENT(...)
    |   LIBRARY = <filename 1..54 without-vers>
    |  ,ELEMENT = <composed-name 1..64>(...)
    |      <composed-name 1..64>(...)
    |         |  VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
    |  ,TYPE = S / <alphanum-name 1..8>
,DIALOG-SELECTION = *NO / *YES
,CONTINUE-AFTER-ERROR = *YES / *NO
```

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
2	0	SDP2000	Warning: Not all elements in the input list could be successfully processed. Guaranteed message: SDP2000
	1	SDP2001	None of the elements in the input list could be successfully processed. Guaranteed message: SDP2001

REPEAT-STMT
Repeat a statement

The REPEAT-STMT command allows you to repeat the execution of a statement (SDF format). During the repetition of the statement, special parts of the statement (name part, operands, operand values) are substituted one after the other by the elements of an input list. These elements can be specified as records in a file, a library element or as list elements of an S variable, or they can be entered directly at the terminal. The statement specified will be called for each element in the input list.

REPEAT-STMT	Alias: **REPSTMT**
STMT = <text 0..1800 with-low>	
,**SUBST**ITUTION**-LIST** = ***TERMINAL** / <filename 1..54 without-gen-vers> / ***VAR**IABLE(...) / ***LIB**RARY**-ELEM**ENT(...)	
***VAR**IABLE(...)	
⎮ **VAR**IABLE**-NAME** = <composed-name 1..255>	
***LIB**RARY**-ELEM**ENT(...)	
⎮ **LIB**RARY = <filename 1..54 without-vers>	
⎮ ,**ELEM**ENT = <composed-name 1..64>(...)	
⎮ <composed-name 1..64>(...)	
⎮ ⎮ **VERS**ION = <u>***HIGH**EST**-EXIST**ING</u> / <composed-name 1..24>	
⎮ ,**TYPE** = <u>**S**</u> / <alphanum-name 1..8>	
,**DIALOG-SEL**ECTION = <u>***NO**</u> / ***Y**ES	
,**CONT**INUE**-AFTER-ERROR** = <u>***Y**ES</u> / ***NO**	

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
2	0	SDP2000	Warning: Not all elements in the input list could be successfully processed. Guaranteed message: SDP2000
	1	SDP2001	None of the elements in the input list could be successfully processed. Guaranteed message: SDP2001

SAVE-RETURNCODE
Save current command return code

SAVE-RETURNCODE saves the current command return code so that it can be evaluated with SDF-P functions. The predefined functions SUBCODE1( ), SUBCODE2( ) and MAINCODE( ) are available for evaluating the return code.

SAVE-RETURNCODE

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

SAVE-VARIABLE-CONTAINER
Save variable container

The SAVE-VARIABLE-CONTAINER command is used to save variable containers.

SAVE-VARIABLE-**CONTAIN**ER

CONTAINER-**NAME** = <composed-name 1..64 with-wild(80)>(...) /
list-poss(2000):<composed-name 1..64>(...)

<composed-name 1..64 with-wild(80)>(...)

| **ELEM**ENT-**VERSION** = ***SAME** / ***INCREMENT**

list-poss(2000):<composed-name 1..64>(...)

| **ELEM**ENT-**VERSION** = ***SAME** / ***INCREMENT**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

SELECT-VARIABLE-ELEMENTS
Select elements from list variable

The SELECT-VARIABLE-ELEMENTS command displays the elements of an S variable (list variable) on the screen and writes any elements selected from these to another S variable.

SELECT**-VAR**IABLE**-ELEM**ENTS

FROM-VARIABLE = <composed-name 1..255>
,**TO-VAR**IABLE = <composed-name 1..255>(...)
    <composed-name 1..255>(...)
        ⎪ **SEL**ECTION**-CODE** = ***NO** / ***Y**ES
,**HEADER-LINE** = ***NONE** / <c-string 1..80>
,**MESSAGE** = ***NONE** / <text 1..240 with-low>
,**DISP**LAYED**-ELEM**ENTS = ***STD** / list-possible(5): <composed-name 1..255>(...)
    <composed-name 1..255>(...)
        ⎪ **LENGTH** = ***BY-VALUE** / <integer 1..75>
        ⎪ ,**TITLE** = ***BY-ELEMENT-NAME** / <c-string 1..75>

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error Guaranteed messages: SDP1120, SPD1121, SPD1122
	64	SPD0259	Operation aborted: selection ignored
	130	SDP0099	No further address space available

SEND-DATA
Transfer data record to program

SEND-DATA should always be used when data records and commands are to be mixed. If a SEND-DATA command occurs in the “data stream”, an EOF condition is not activated implicitly, as for other commands; instead, this is controlled by the RECORD operand.

SEND-DATA
RECORD = ***EOF** / <text 0..1800 with-low *string-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

SEND-STMT
Transfer statement record to program

SEND-STMT should always be used when commands and statements are mixed. Just like the SEND-DATA command, a SEND-STMT command in the data stream does not implicitly activate an EOF condition

SEND-STMT
RECORD = ***EOF** / <text 0..1800 with-low *string-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

SET-PROCEDURE-OPTIONS
Set procedure attributes

The SET-PROCEDURE-OPTIONS command serves to define the attributes of an *S procedure*. The command is optional. If used, however, it must be the *first* command in the procedure head. If not used, the attributes according to the presettings of SDF-P are valid.

SET-PROCEDURE-OPTIONS
CALLER = *ANY / *CALL / *INCLUDE
,IMPLICIT-DECLARATION = *YES / *NO
,LOGGING-ALLOWED = *PARAMETERS(...) / *YES / *NO
*PARAMETERS(...)
CMD = *YES / *NO
,DATA = *YES / *NO
,INTERRUPT-ALLOWED = *YES / *NO
,INPUT-FORMAT = *FREE-RECORD-LENGTH / *BY-SDF-OPTION
,DATA-ESCAPE-CHAR = *NONE / '&&' / '#' / '*' / '@' / '$' / *STD
,SYSTEM-FILE-CONTEXT = *STD / *SAME-AS-CALLER / *OWN
,DATA-ERROR-HANDLING = *YES / *NO
,JV-REPLACEMENT = *NONE / *AFTER-BUILTIN-FUNCTION
,ERROR-MECHANISM = *SPIN-OFF-COMPATIBLE / *BY-RETURNCODE
,SUPPRESS-SDP-MSG = *NONE / list-poss(2000): <alphanum-name 7..7>

Return codes

The following return codes can thus appear only if SET-PROCEDURE-OPTIONS is used outside the procedure head.

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	130	SDP0099	No further address space available

SET-VARIABLE
Assign value to variable

SET-VARIABLE can be used to assign values to simple as well as complex variables.

When assigning simple variables or variable elements, the data types must match. The structure of the variables must also match when assigning complex variables globally.

SET-VARIABLE	Alias: **STV**
<composed-name1 1..255> = <text 0..1800 with-low *expr*> / <composed-name2 1..255> **/** ***STRING-TO-VAR**IABLE(...) / ***LIST**(...) ***STRING-TO-VAR**IABLE(...**)** ⏐ **STRING** = <text 0..1800 with-low *expr*> ⏐ ,**VALUE-TYPE** = ***STD** / ***STRING** ***LIST(...)** ⏐ **LIST-NAME** = <composed-name 1..255> ⏐ ,**FROM-INDEX** = ***FIRST** / ***LAST** / <integer 1..2147483647> ⏐ ,**NUMBER-OF-ELEM**ENTS = **1** / ***REST** / <integer 1..2147483647> **,WRITE-MODE** = ***REPLAC**E / ***MERGE** / ***EXTEND** / ***PREFIX**	

Return codes

During the assignment of structures, arrays or lists, it is possible that part of the command has been processed and executed when the error occurs. In this case, the result of the command is not guaranteed.

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error Guaranteed messages: SDP1030
	130	SDP0099	No further address space available

SHOW-STREAM-ASSIGNMENT
Show S variable stream

SHOW-STREAM-ASSIGNMENT shows the current assignment of the specified S variable streams.

SHOW-STREAM-ASSIGNMENT
STREAM-NAME = ***ALL** / ***STD-STREAMS** / <structured-name 1..20 with-wild(40)> / list-poss(100): <structured-name 1..20> ,**INF**ORMATION = ***CURR**ENT**-ASSIGNMENT** / ***FINAL-DEST**INATION ,**OUTPUT** = ***SYSOUT** / ***SYSLST**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	32	CMD2009	Error during S variable replacement
	64	CMD0216	Do not have required privilege
	64	OPS0001	No storage for S variables
	64	SDP0091	Semantic error
	64	SDP0517	Specified variable steam name does not exist
	64	SDP0519	No match for specified wildcards

SHOW-STRUCTURE-LAYOUT
Output element name of structure layout

SHOW-STRUCTURE-LAYOUT outputs element names for a particular structure layout. The structure layout must have been defined beforehand using BEGIN-STRUCTURE.

SHOW-STRUCTURE-**LAYOUT**	Alias: **SHSTRL**

```
NAME = *ALL / <structured-name 1..20 with-wild(40)> / list-poss(2000): <structured-name 1..20>
,SCOPE = *VISIBLE / *PROCEDURE / *CURRENT / *TASK
,OUTPUT = *SYSOUT / *SYSLST / <filename 1..54 without-gen-vers>(...) / *VARIABLE(...) /
          *LIBRARY-ELEMENT(...)
   <filename 1..54 without-gen-vers>(...)
     | WRITE-MODE = *REPLACE / *EXTEND
   *VARIABLE(...)
     | VARIABLE-NAME = <composed-name 1..20>
     | ,WRITE-MODE = *REPLACE / *EXTEND
   *LIBRARY-ELEMENT(...)
     | LIBRARY = <filename 1..54 without-vers>
     | ,ELEMENT = <composed-name 1..64>(...)
     |    <composed-name 1..64>(...)
     |      | VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24>
     | ,TYPE = S / <alphanum-name 1..8>
```

Return codes

It is possible that part of the command has been processed and executed when the error occurs. In this case, the result of the command is not guaranteed..

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
1	0	CMD0001	Warning: no layout found
2	0	SDP2000	Warning: not all elements of the input list could be processed successfully. Guaranteed message: SDP2000
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	64	SDP1008	Variable does not exist
	64	SDP2001	None of the elements could be displayed
	130	SDP0099	No further address space available

SHOW-VARIABLE
Output contents of variables

SHOW-VARIABLE outputs the contents of variables.
If SHOW-VARIABLE is applied to a list or an array, the elements are output in the numeric order of their indices.
If SHOW-VARIABLE is applied to a structure, the elements are output in the order of the element declarations.

```
SHOW-VARIABLE                                                                    Alias: SHV

VARIABLE-NAME = *ALL / *LIST(...) /
             list-poss(2000): <composed-name 1..255> /<structured-name 1..20 with-wild(40)>
  *LIST(...)
    |  LIST-NAME = <composed-name 1..255>
    | ,FROM-INDEX = *FIRST / *LAST / <integer 1..2147483647>
    | ,NUMBER-OF-ELEMENTS = 1 / *REST / <integer 1..2147483647>
,SELECT = *BY-ATTRIBUTES(...)
  *BY-ATTRIBUTES(...)
    |  SCOPE = *VISIBLE / *PROCEDURE / *CURRENT / *CURRENT-PARAMETERS / *TASK-VISIBLE /
    |                 *TASK-ALL / *CALLING-PROCEDURES
    | ,INITIALIZATION = *YES / *ANY
,INFORMATION = *PARAMETERS(...)
  *PARAMETERS(...)
    |  VALUE = *WITHOUT-QUOTES / *C-LITERAL / *X-LITERAL / *NONE
    | ,NAME = *FULL-NAME (...) / *ELEMENT-NAME (...) / *NONE
    |    *FULL-NAME(...)
    |      |  LIST-INDEX-NUMBER = *NO / *YES
    |    *ELEMENT-NAME(..)
    |      |  LIST-INDEX-NUMBER = *NO / *YES
,OUTPUT = *SYSOUT / *SYSLST / <filename 1..54 without-gen-vers>(...) / *VARIABLE(...) /
            *LIBRARY-ELEMENT(...)
  <filename 1..54 without-gen-vers>(...)
    |  WRITE-MODE = *REPLACE / *EXTEND
  *VARIABLE(...)
    |  VARIABLE-NAME = <composed-name 1..20>
    | ,WRITE-MODE = *REPLACE / *EXTEND
  *LIBRARY-ELEMENT(...)
    |  LIBRARY = <filename 1..54 without-vers>
    | ,ELEMENT = <composed-name 1..64>(...)
    |    <composed-name 1..64>(...)
    |      |  VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24>
    | ,TYPE = S / <alphanum-name 1..8>
    | ,WRITE-MODE = *REPLACE / *EXTEND
```

Return codes

It is possible that part of the command has been processed and executed when the error occurs. In this case, the result of the command is not guaranteed.

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Warning; no variable found
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error Guaranteed message: SDP1008
	130	SDP0099	No further address space available

SHOW-VARIABLE-ATTRIBUTES
Output variable attributes

The SHOW-VARIABLE-ATTRIBUTES command supplies information about the attributes of the specified variables. The attributes include the name of the variable, its initial value, data type, a simple or a complex variable, etc.

SHOW-VARIABLE-ATTRIBUTES
VARIABLE-NAME = *ALL / <composed-name 1..255> / <structured-name 1..20 with-wild(40)> / *LIST(...)
*LIST(...)
LIST-NAME = <composed-name 1..255>
,FROM-INDEX = *FIRST / *LAST / <integer 1..2147483647>
,NUMBER-OF-ELEMENTS = 1 / *REST / <integer 1..2147483647>
,INFORMATION = *NAME / *VARIABLE-ATTRIBUTES-ONLY / *ALL-ATTRIBUTES
,ATTACHED-INFORMATION = *NO / *YES
,OUTPUT = *SYSOUT / *SYSLST

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error (variable does not exist) Guaranteed message: SDP1008
	130	SDP0099	No further address space available

SHOW-VARIABLE-CONTAINER-ATTR
Display open variable containers

The SHOW-VARIABLE-CONTAINER-ATTR command displays all the open variable containers.

SHOW-VARIABLE-CONTAINER-ATTR
CONTAINER-NAME = ***ALL** / <composed-name 1..64 with-wild(80)> / list-poss: <composed-name 1..64>
,**CONTAINER-SCOPE** = ***VISIBLE** / ***PROCEDURE** / ***CURRENT** / ***TASK**
,**OUTPUT** = ***SYSOUT** / ***SYSLST**

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

SORT-VARIABLE
Sort list variable

The SORT-VARIABLE command sorts the elements of a list variable in ascending or descending order according to their content. Only list variables whose elements are simple variables of the same type (STRING, INTEGER, BOOLEAN oder ANY) can be sorted.

SORT-VARIABLE
VARIABLE-NAME = list-poss(2000): <composed-name 1..255>
,**SORTING-ORDER** = ***ASCENDING** / ***DESCENDING**

Command return code

When an error occurs and only some of the commands have been processed and executed, the result of the command is not guaranteed.

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	SDP2000	Warning: some elements could not be sorted
	64	SDP2001	None of the elements could be sorted

TRACE-PROCEDURE
Resume interrupted procedure in stages

TRACE-PROCEDURE resumes an interrupted procedure in stages. Commands executed subsequently are only logged if logging is permitted.

TRACE-PROCEDURE	Alias: **TCP**
STEPS = ***LAST-INP**UT / <text 0..1800 with-low *arith-expr*>	

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

TRANSMIT-BY-STREAM
Transmit variables

The TRANSMIT-BY-STREAM command carries out a variable transmission initiated from the client side, to or from the server which is addressed and using the specified S variable stream. If *DUMMY is assigned to the S variable stream, the command sends back a return code indicating that nothing has been changed.

TRANSMIT-BY-STREAM
STREAM-NAME = <structured-name 1..20> ,VARIABLE-NAME = *NONE / <composed-name 1..255> ,RETURN-VARIABLE-NAME = *SAME / *NONE / <composed-name 1..255> ,CONTROL-VAR-NAME = *NONE / <composed-name 1..255> ,RET-CONTROL-VAR-NAME = *SAME / *NONE / <composed-name 1..255>

Return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	No error
1	0	CMD0001	Variable stream is assigned to *DUMMY; no transmission, variable remains unchanged
2	0	SDP0512	Server is no longer active. Data stream is assigned to *DUMMY
2	0	SDP0531	Warning returned by server; process continuing
	1	CMD0202	Syntax error
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	CMD0216	Do not have required privilege
	64	SDP0091	Semantic error Guaranteed messages: SDP1008
	64	SDP0532	Server error; command rejected
	64	SDP0534	Internal server error; command terminated. Server link terminated following unexpected event or due to shortage or absence of system resources
	64	SDP0517	Variable stream does not exist
	64	SDP0522	Transmitted data is incompatible with the format that the server handles
	64	SDP1132	Variable name too long
	130	SDP0099	No further address space available

UNTIL
Terminate REPEAT block

UNTIL contains the loop condition for a REPEAT block, i.e. for a REPEAT loop, and terminates the REPEAT block (the initiation command for the REPEAT block is REPEAT).

UNTIL
CONDITION = <text 0..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0139	Back branch limit reached
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

WHILE
Initiate WHILE block

WHILE initiates a WHILE block, i.e. a WHILE loop. The loop must be terminated using the END-WHILE command.

WHILE
CONDITION = <text 0..1800 with-low *bool-expr*>

Return codes

(SC2)	SC1	Maincode	Meaning
	0	CMD0001	No error
	1	CMD0202	Syntax error
	1	SDP0118	Command in false context
	1	SDP0223	Incorrect environment
	3	CMD2203	Incorrect syntax file
	32	CMD0221	System error (internal error)
	64	SDP0091	Semantic error
	130	SDP0099	No further address space available

Program interface

Overview

System administrator macros	Brief description
BIFDEF	Generates the table entry which links the name of the function to the addresses of the entries of the executable module and the syntax description
BIFDESC	Contains the syntax description of the system administrator functions
BIFMDL1	Generates the DSECT for the values of the system administrator functions
BIFMDL2	Generates the DSECT for the return code of the system administrator functions

User macros	Brief description
CLIEXPR	Evaluates SDF-P expressions
CLIGET	Queries procedure interrupt protection
CLISET	Defines explicit protection against program interrupts
CMD	Issues SDF-P commands from a program
GETVAR	Reads simple and complex variables
PUTVAR	Writes simple and complex variables
SHOWSSA	Shows variable stream assignments
TRANSVV	Transfers variables via a variable stream
VARINF	Processes (modifies) complex variables

Macros

BIFDEF

The BIFDEF macro generates the table entries which link the names of functions to the addresses of the entries for their executable modules and syntax specifications.

Operation	Operands
BIFDEF	MF = L NAME = <name 1..20> SYNTAX = <name 1..8> CODE = <name 1..8>

BIFDESC

The BIFDESC macro contains the syntax specification for system administration functions.

The purpose of this macro is to define a static structure which will be used exclusively by the SDF-P-BIF subsystem.

Operation	Operands
BIFDESC	NAME = <name 1..20> ,ENTRYN = <name 1..8> / (*CSECT,<name 1..8>) ,PARLIST = *NONE / list-poss(2000): (parameter-specification) ,PARFORM = *BY-VALUE / *STRING ,VALTYPE = *STRING / *INTEGER / *BOOLEAN / *ANY

BIFMDL1

The BIFMDL1 macro generates a DSECT for the values of the system administration functions. The structure of each element in the operand list for the executable module is described.

Operation	Operands
BIFMDL1	MF = D ,PREFIX = B / prefix ,MACID = IF1 / macid

BIFMDL2

The BIFMDL2 macro generates a DSECT for the return code from the system administration functions, or contains the specification of the return code which is supplied by the executable module.

Operation	Operands
BIFMDL2	MF = D ,PREFIX = B / prefix ,MACID = IF2 / macid

CLIEXPR

The CLIEXPR macro evaluates arithmetic, logical and string expressions. The expression is passed in an input field, the result is returned in an output field. It is possible to request a specific output format for the result (binary number, Boolean constant, string).

The macro can also be called using MF = M.

Operation	Operands
CLIEXPR	MF = E ,PARAM = <name 1..27> / (<integer 1..15>)
	MF = D [,PREFIX = C / prefix]
	MF = C ,PREFIX = C / prefix [,MACID = LIE / macid]
	MF = L ,INPUT@ = <pointer> ,INPUTL = <integer 0..2147483647> ,OUTPUT@ = <pointer> ,OUTPUTL = <integer 0..2147483647> ,VFORM = *BY-VALUE / *STRING ,OTYPE = <pointer> ,OACTL = <pointer> ,PROT@ = NULL / <pointer> ,PROTL = 0 / <integer 0..2147483647> ,OPROTL = NULL / <pointer>

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Normal execution
01	00	0000	Overflow of PROT field (Warning)
00	40	0001	Syntax error in expression to be evaluated
01	40	0001	Overflow of PROT field
00	40	0002	Error during evaluation
01	40	0002	Overflow of PROT field
00	40	0003	Output field too short
00	01	0004	Input field not specified or not aligned
01	01	0004	Output field not specified or not aligned
02	01	0004	Log field (not aligned)
03	01	0004	Other fields (not aligned)
04	01	0004	Field address specified but field not accessible
00	40	0005	Insufficient space in caller’s address space
01	20	0006	System error
00	40	0007	Invalid procedure format; macro execution has been aborted
00	01	FFFF	Wrong specification for UNIT or FUNCTION in standard header
00	02	FFFF	Requested function is not supported
00	03	FFFF	Wrong version specification in standard header

CLIGET

The CLIGET macro enables a program to query whether protection is required against implicit interruptions.
CLIGET returns the setting of the INTERRUPT-ALLOWED operand, as specified in the SET-PROCEDURE-OPTIONS, MODIFY-PROCEDURE-OPTIONS or BEGIN-PROCEDURE commands.

Operation	Operands
CLIGET	MF = E ,PARAM = <name 1..8> / (integer 1..15)
	MF = D [,PREFIX = C / prefix]
	MF = C [,PREFIX = C / prefix] [,MACID = LIS / macid]
	MF = L

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Macro call was successful; no error
00	01	0001	Parameter error; parameter too short
00	20	0004	System error
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of the operand list

CLISET

The CLISET macro explicitly protects programs against interruptions.

Operation	Operands
CLISET	MF = E ,PARAM = <name 1..8> / (integer 1..15)
	MF = D [,PREFIX = C / prefix]
	MF = C [,PREFIX = C / prefix] [,MACID = LIS / macid]
	MF = L [,EXNINT = *U / *Y/ *N]

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
01	00	0000	No action (EXNINT=*U)
00	00	0000	Macro call was successful; no error
00	01	0001	Parameter error
01	00	0002	EXNINT=*Y already set previously, or EXNINT=*N already set previously
00	20	0004	System error
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of the operand lis

CMD

The CMD macro call can be used to execute commands, including SDF-P commands, in assembler programs.

Command	Function
ASSIGN-STREAM	Assign S variable stream
BEGIN-STRUCTURE	Start static structure declaration
CALL-PROCEDURE	Start procedure
CLOSE-VARIABLE-CONTAINER	Close variable container
DECLARE-CONSTANT	Declare variable with constant value
DECLARE-ELEMENT	Declare structure element
DECLARE-VARIABLE	Declare variable
DELETE-STREAM	Delete S variable stream
DELETE-VARIABLE	Delete variable
END-STRUCTURE	Identify end of structure declaration
ENTER-PROCEDURE	Start procedure as background procedure
FREE-VARIABLE	Delete variable contents
IMPORT-VARIABLE	Import variable
INCLUDE-CMD	Pass command sequence
INCLUDE-PROCEDURE	Start INCLUDE procedure
MODIFY-PROCEDURE-OPTIONS	Modify procedure attributes
OPEN-VARIABLE-CONTAINER	Open variable container
READ-VARIABLE	Read in variable values
SAVE-VARIABLE-CONTAINER	Save variable container
SELECT-VARIABLE-ELEMENTS	Select elements of a list variable
SHOW-STREAM-ASSIGNMENT	Display assignment for S variable stream
SHOW-STRUCTURE-LAYOUT	Display element name of structure layout
SHOW-VARIABLE	Display variable contents
SHOW-VARIABLE-ATTRIBUTES	Display variable attributes
SHOW-VARIABLE-CONTAINER-ATTR	Display variable container attributes
SORT-VARIABLE	Sorts the elements of a list variable
TRANSMIT-BY-STREAM	Transmit variables with S variable stream

GETVAR

The GETVAR macro reads the contents of a variable.
GETVAR can be used with simple variables and elements of complex variables.

Operation	Operands
GETVAR	MF = E ,PARAM = <name 1..8> / (<integer 1..15>)
	MF = D ,PREFIX = G / prefix
	MF = C ,PREFIX = G / prefix ,MACID = ETV / macid
	MF = L ,NAMLEN = <integer 1..255> ,NAMADR = <name 1..8> ,SCOPE = *VISIBLE / *TASKONLY ,MAXLEN = <integer 1..4096> ,VALADR = <name 1..8>

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Macro call was successful; no errors
00	01	0001	Parameter error
00	01	0002	Syntax error in variable name
00	40	0003	Area too small
00	40	0004	Variable not declared
00	40	0005	Variable container not available
00	40	0006	Data type and variable value do not match
00	40	0008	Variable has no value
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of operand list

PUTVAR

The PUTVAR macro assigns a value to a variable. The variable may be a simple variable or an element of a complex variable.
If the assignment refers to a simple variable which does not yet exist, it is created either if IMPLICIT-DECLARATION=YES and IMPDEC = *STD apply, or if the macro call specifies IMPDEC=*YES.

Operation	Operands
PUTVAR	MF = E
	,PARAM = <name 1..8> / (<integer 1..15>)
	MF = D
	,PREFIX = P / prefix
	MF = C
	,PREFIX = P / prefix
	,MACID = UTV / macid
	MF = L
	,NAMLEN = <integer 1..255>
	,NAMADR = <name 1..8>
	,SCOPE = *VISIBLE / *TASKONLY
	,IMPDEC = *YES / *NO / *STD
	,VALLEN = <integer 0..4096>
	,VALADR = <name 1..8>
	,VALTYPE = *INTEGER / *BOOLEAN / *STRING

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Macro call was successful; no errors
00	01	0001	Parameter error
00	01	0002	Syntax error in variable name
00	40	0004	Variable not declared
00	40	0005	Variable container not available
00	40	0006	Data type and variable value do not match
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of operand list

SHOWSSA

The SHOWSSA macro shows the current assignment of the specified S variable stream. It is functionally equivalent to the SHOW-STREAM-ASSIGNMENT command. However, it is not possible to specify a list of streams with SHOWSSA.

Operation	Operands
SHOWSSA	MF = E ,PARAM = <name 1..8> / (<integer 1..15>)
	MF = D ,PREFIX = S / prefix
	MF = C ,PREFIX = S / prefix ,MACID = HOW / macid
	MF = L / M ,STREAM = *ALL / *STD_STREAMS / <c-string 1..20 with-wild> ,INFO = *ASSIGNMENT / *DESTINATION ,OUTPUT = *RETURN–CODE / *SYSOUT / *SYSLST

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Macro call was successful; no errors
00	01	0001	Parameter error
00	40	0002	Specified variable stream is incomplete
00	40	0003	SSTA error (SSTA too small, not initialized, ...)
00	40	0004	Terminated by K2 during output to SYSOUT
00	40	0005	Error during output to SYSOUT
00	40	0006	Error during output to SYSLST
02	01	0007	More than one variable stream for OUTPUT=*RETURNCODE
00	20	0008	System error
02	00	000A	Specified variable stream is assigned to *DUMMY
02	00	000B	Specified variable stream is already assigned
02	00	000C	Specified variable stream does not exist
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of the operand list
00	41	FFFF	SDF-P is not loaded
00	81	FFFF	SDF-P no longer working

TRANSVV

The TRANSVV macro is used by a client to carry out a variable transmission via the specified S variable stream to the server that is currently assigned (ASSIGN-STREAM command). TRANSVV is functionally equivalent to the TRANSMIT-BY-STREAM command. TRANSVV can only use S variable streams that were assigned at the same procedural hierarchy level at which the program was started.

Operation	**Operands**
TRANSVV	MF = E
	,PARAM = <name 1..8> / (<integer 1..15>)
	MF = D
	[,PREFIX = T / prefix]
	MF = C
	[,PREFIX = T / prefix]
	[,MACID = RAN / macid]
	MF = L/M
	,STREAM = <name 1..20>
	[,VNAME = *NONE / <name 1..8> / (integer 1..15)]
	[,VNAMEL = <integer 1..255>]
	[,VSCOPE = *VISIBLE / *TASKONLY]
	[,RNAME = *SAME / *NONE / <name 1..8> / (integer 1..15)]
	[,RNAMEL = <integer 1..255>]
	[,RSCOPE = *VISIBLE / *TASKONLY]
	[,CNAME = *NONE / <name 1..8> / (integer 1..15)]
	[,CNAMEL = <integer 1..255>]
	[,CSCOPE = *VISIBLE / *TASKONLY]
	[,RCNAME = *SAME / *NONE / <name 1..8> / (integer 1..15)]
	[,RCNAMEL = <integer 1..255>]
	[,RCSCOPE = *VISIBLE / *TASKONLY]

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Transmission successfully completed; no error
01	00	0000	Variable stream was assigned to *DUMMY, no transmission
00	01	0001	Parameter error
00	40	0002	Specified variable stream is incomplete
00	40	0003	Specified variable is incomplete
00	40	0004	RET-SSTA too small (for developers only)
00	01	0005	The data items transmitted (user or control data) do not have a format compatible with one the server can process
00	40	0006	Error message from the server; saved in RCNAME (if specified)
02	00	0007	Warning from the server; saved in RCNAME (if specified)
02	00	0008	Variable stream reset to *DUMMY; server is no longer active
00	20	0009	System error
00	20	000A	Error during server connection
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function is not available
00	03	FFFF	Wrong version of the operand list
00	41	FFFF	SDF-P is not loaded
00	81	FFFF	SDF-P no longer working

VARINF

The VARINF macro can be used to analyze complex variables whose elements are themselves complex variables.

Operation	Operands
VARINF	MF = E ,PARAM = <name 1..8> / (<integer 1..15>)
	MF = D ,PREFIX = V / prefix
	MF = C ,PREFIX = V / prefix ,MACID = ARI / macid
	MF = L ,NAMLEN = <integer 1..255> ,NAMADR = <name 1..8> ,SCOPE = *VISIBLE / *TASKONLY ,POSIT = *CURRENT / *UP / *DOWN / *NEXT ,MAXLEN = <integer 1..4096> ,RESADR = <name 1..8>

Return codes (hexadecimal)

Subcode2	Subcode1	Maincode	Meaning
00	00	0000	Macro call was successful; no error
00	01	0001	Parameter error
00	01	0002	Syntax error in variable name
00	40	0003	Area too small
00	40	0004	Variable not declared
00	40	0005	Variable container not available
00	40	0007	Last variable element was reached; no further variable elements present
00	01	FFFF	Unknown unit or function number
00	02	FFFF	Function not available
00	03	FFFF	Wrong version of operand list

Formation of variable names

The SDF data types <composed-name> and <structured-name> are used for variable names and partial variable names in SDF-P in the command syntax. However, the syntax checked by SDF-P is restricted in comparison with these data types and is therefore described here in more detail.

A variable name (composed-variable-name) in SDF-P consists of one or more structure names (structured-variable-name) separated from each other by periods. The period in each case marks the name of a substructure. Blanks are not permitted within a variable name.

```
composed-variable-name:=structured-variable-name[.structured-variable-name]
```

Length	at least 1 character, not more than 255 characters

A structure name consists of a partial variable name (pvn) which, if it refers to a list element or an array element, may be followed by a # and an index name.

```
structured-variable-name := pvn [# [indexname]]
```

Length	at least 1 character, not more than 20 characters
Partial variable	name (pvn):
Character set	all letters (A, ... Z) all digits (0, ... 9) @ and hyphen (-)
First charactern	a letter
Conventions	The string SYS at the beginning of a variable name is reserved for system variables, and should not be used. A hyphen may not be directly followed by another hyphen. The hyphen must not be the last character of the partial variable name..

If the structure name refers to a list element or an array element, it is followed by a # and an index name.

```
indexname = <integer -2147483648..2147483647>  / pvn
```

Element names

In the case of element names, a distinction must be made between array elements, list elements and structure elements.

The syntax that is described under <composed-variable-name> also applies to element names.

List element names

List element names consist, at the user interface, of the following components:

– list name (<composed-variable-name 1..253>)
– #
– optional: element number (<integer 1..2147483647>)

Array element names

Array element names are made up of the following components:

– array name (<composed-variable-name 1..253>)
– # (identifier for array elements when followed by additional characters)
– array index (<integer -2147483648..2147483647>)

Both the list index and the array index can also be identified by a simple variable that contains an integer value within the range of valid values.

SDF-P analyzes the string that follows the aggregate symbol # in the element name: If the first character is a digit or hyphen (= minus sign), the string is interpreted as an integer value, i.e. as a direct entry of the index.

If the first character is a letter, the string is interpreted as a variable name. SDF-P then searches for a variable with the specified name. This name must refer to a simple variable, which must be initialized with a valid integer value. If this is not the case, an error message is issued. If the variable contains a valid integer value, this value is used in the array index or list index, and the aggregate element thus identified is accessed.

Struktur element names

Structure element names are made up of the following components:

– name of the structure (<composed-variable-name 1..253>)
– "." (period: ID for structures)
– subname of the element (<structured-variable-name 1..20>)

Reserved words

"Reserved words" are keywords used as operators in expressions or Boolean constants. They must not be declared as variable names.

If a reserved word is nevertheless used to declare a variable, an error message will be issued and the variable will not be generated.

The following are reserved words:

AND	LE	OFF
DIV	LT	ON
EQ	MOD	OR
FALSE	NE	TRUE
GE	NO	YES
GT	NOT	

Reserved variable names

Variable names which begin with the prefix SYS are reserved for transferring data to and from system components. Here, the variable name SYSSWITCH is of particular significance.

SYSSWITCH

The variable name SYSSWITCH identifies a complex variable of type array, which can be used to address the job switch.

The SYSSWITCH array is defined as follows:

Data type	BOOLEAN
Scope	TASK (in a dialog and procedure environment)
Number of elements	32
Array index	0,..., 31
Values	TRUE = switch set to ON, FALSE = switch set to OFF

The array elements can be addressed by the partial names SYSSWITCH#0 to SYSSWITCH#31. Both the read mode and the write mode are permitted.

Neither the SYSSWITCH array nor the array elements can be deleted; i.e. they cannot be specified in a FREE-VARIABLE- or DELETE-VARIABLE command.

The SYSSWITCH array is always implicitly declared in every procedure.

SYSPARAM

The variable name SYSPARAM designates a variable of type string that can be used to access the program parameters passed with the START-/LOAD-EXECUTABLE-PROGRAM commands. In C programs the program parameters are accessed with the `getopt` function, and assembler programs must read in the SYSPARAM variable and evaluate it themselves using the GETVAR macro call (see page 118).

Metasyntax, data types, operators

Metasyntax for commands and macros

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords (command, statement, operand names; keyword values) and constant operand values. Keyword values begin with *.	**HELP-SDF** **SCREEN-STEPS** = <u>***NO**</u>
UPPERCASE LETTERS in boldface	Uppercase letters printed in boldface denote guaranteed or suggested abbreviations of keywords.	**GUID**ANCE-**MODE** = ***Y**ES
=	The equals sign connects an operand name with the associated operand values.	**GUID**ANCE-**MODE** = <u>***NO**</u>
< >	Angle brackets denote variables whose range of values is described by data types and suffixes.	**SYNTAX-F**ILE = <filename 1..54>
<u>Underscoring</u>	Underscoring denotes the default value of an operand.	**GUID**ANCE-**MODE** = <u>***NO**</u>
/	A slash serves to separate alternative operand values.	**NEXT-FIELD** = <u>***NO**</u> / ***Y**ES
(…)	Parentheses denote operand values that initiate a structure.	,**UNGUID**ED-**DIA**LOG = <u>***Y**ES</u> (...) / ***NO**
[ ]	Square brackets denote operand values which introduce a structure and are optional. The subsequent structure can be specified without the initiating operand value.	**SELECT** = [***BY-ATTR**IBUTES](...)

Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	,**GUID**ED-**DIA**LOG = <u>***Y**ES</u> (...)
		  <u>***Y**ES</u>(...)
		    | **SCREEN-STEPS** = <u>***NO**</u> /
		    | ***Y**ES

|	A vertical bar identifies related operands within a structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	**SUP**PORT = ***TAPE**(...)
		  ***TAPE(...)**
		    | **VOL**UME = <u>***ANY**</u>(...)
		    | <u>***ANY**</u>(...)
		    | | ...

Representation	**Meaning**	**Examples**
,	A comma precedes further operands at the same structure level.	**GUID**ANCE**-MODE** = __*NO__ / ***Y**ES ,**SDF-COM**MANDS = __*NO__ / ***Y**ES
list-poss(n):	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <filename 1..54>
Alias	The name that follows represents a guaranteed alias for the command or statement name.	**HELP-SDF** Kurzname: **HPSDF**

Metasyntax for functions

Representation	Meaning	Examples
UPPERCASE LETTERS	Designate keywords; some keywords begin with an *	POSITION = *FIRST
=	Links an operand name to its operand values	STREAM = *COMMAND / *DATA
	Separates alternative operand values	STREAM = *COMMAND / *DATA
<u>Underscoring</u>	Designates the default value of an operand	DIRECTION = **<u>*FORWARD</u>** / RESERVE
expression	Any expression	**EXPRESSION = expression**
string_expression	Expression that returns a string	STRING = string_expression
arithm_expression	Expression that returns a numeric value	INTEGER = arithm_expression
integer	Integer (integer value)	START = <u>1</u> / zahl
character	Character in EBCDI code	FILL-BYTE = character
msg_id	Seven-character message code	MSD-IDENTIFICATION = msg_id

Metasyntax for variable names

Representation	Meaning	Examples
:=	Definition	varname:=
<>	Data type (see section entitled “Data types”)	<composed-name 1...255>
m..n	Value range (in the data type) Repetition	1..255
[ ]	Optional entry	[<symbol>]
/	Alternative entries	

Data types

Data types	Character set	Special rules
alphanum-name	A…Z 0…9 $, #, @	
cat-id	A…Z 0…9	Not more than 4 characters; must not begin with the string PUB
command-rest	Freely selectable	
composed-name	A…Z 0…9 $, #, @ hyphen period catalog ID	Alphanumeric string that can be split into multiple substrings by means of a period or hyphen. If a file name can also be specified, the string may begin with a catalog ID in the form :cat: (see data type filename below).
c-string	EBCDIC character	Must be enclosed within single quotes; the letter C may be prefixed; any single quotes occurring within the string must be entered twice.
date	0…9 Structure identifier: hyphen	Input format: jjjj-mm-tt jjjj: year; optionally 2- or 4 digits mm: month tt: day
device	A…Z 0…9 hyphen	Character string, max. 8 characters in length, corresponding to a device available in the system. In guided dialog, SDF displays the valid operand values. For notes on possible devices, see the relevant operand description.
fixed	+, - 0…9 period	Input format: [character][ziffern].[ziffern] [sign]: + oder - [digits]: 0...9 must contain at least one digit, and may contain up to 10 characters (0...9, period) apart from the sign.

Data types	**Character set**	**Special rules**
filename	A…Z 0…9 $, #, @ hyphen period	Input format: $[:cat:][\$user.]\left\{\begin{array}{l} \text{file} \\ \text{file(no)} \\ \text{group} \\ \text{group}\left\{\begin{array}{l}\text{(*abs)}\\ \text{(+rel)}\\ \text{(-rel)}\end{array}\right\} \end{array}\right\}$:cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed in colons; the default value is the catalog identifier assigned to the user ID, as specified in the user catalog. $user. optional entry of the user ID; character set is A…Z, 0…9, $, #, @; maximum of 8 characters; first character must not be a digit; $ and period are mandatory; the default value is the user' s own ID. $. (special case) system default ID file file or job variable name; may be split into a number of partial names using a period as a delimiter: name$_1$[.name$_2$[...]] name$_i$ does not contain a period and must not begin or end with a hyphen; file can have a max. length of 41 characters; it must not begin with a $ and must include at least one character from the range A...Z. #file (special case) @file (special case) # or @ used as the first character indicates temporary files or job variables, depending on system generation.

Data types	Character set	Special rules
filename (continued)		file(no) Tape file name no: version number; character set is A…Z, 0…9, $, #, @. Parentheses must be specified. group Name of a file generation group (character set: as for "file") group { (*abs) (+rel) (-rel) } (*abs) absolute generation number (1..9999); * and parentheses must be specified. (+rel) (-rel) relative generation number (0..99); sign and parentheses must be specified.
integer	0…9, +, -	+ or -, if specified, must be the first character (sign).
name	A…Z 0…9 $, #, @	Must not begin with a digit
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see filename $user. see filename partname optional entry of the initial part of a name common to a number of files or file generation groups in the form: $name_1$.[$name_2$.[...]] $name_i$ (see filename). The final character of “partname” must be a period. At least one of the parts :cat:, $user. or partname must be specified.

Data types	**Character set**	**Special rules**
posix-filename	A...Z 0...9 special characters	String with a length of up to 255 characters; consists of either one or two periods or of alphanumeric characters and special characters. The special characters must be escaped with a preceding \ (backslash); the slash (/) is not allowed. Must be enclosed within single quotes if alternative data types are permitted, if separators are used, or if the first character is a ?, ! or ^. A distinction is made between uppercase and lowercase.
posix-pathname	A...Z 0...9 special charactersn structure identifier: slash	Input format: [/]part$_1$/.../part$_n$ where part$_n$ is a posix-filename; max. 1024 characters; must be enclosed within single quotes if alternative data types are permitted, if separators are used, or if the first character is ?, ! or ^.
product-version	A…Z 0…9 period single quote	Input format: [[C]'][V][m]m.naso['] correction status release status where m, n, s and o are all digits and a is a letter. Whether specification of the release and/or correction status is optional or mandatory is determined by suffixes to the data type (see the corresponding table: suffixes without-corr, without-man, mandatory-man and mandatory-corr). product-version may be enclosed in single quotes (possibly with a preceding C). The version specification may begin with the letter V.
structured-name	A…Z 0…9 $, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A…Z or $, #, @
text	Freely selectable	For the input format, see the relevant operand descriptions.

Data types	**Character set**	**Special rules**
time	0…9 structure identifier: colon	Time-of-day entry Input format: { hh:mm:ss / hh:mm / hh } hh: hours mm: minutes ss: seconds Leading zeros may be omitted
vsn	a) A…Z 0…9 b) A…Z 0…9 $, #, @	a) Input format: pvsid.sequence-no max. 6 characters; pvsid: 2-4 characters; PUB is not permitted sequence-no: 1 1-3 characters b) max. 6 characters; PUB may be prefixed, but must not be followed by $,#,@.
x-string	Hexadecimal: 00…FF	Must be enclosed in single quotes; must be prefixed by the letter X. There may be an odd number of characters.
x-text	Hexadecimal: 00…FF	Must not be enclosed in single quotes; the letter X must not be prefixed. There may be an odd number of characters.

Suffixes for data types

Suffix	Meaning
x..y *unit*	with data type integer: interval specification x — minimum value permitted for “integer”. x is an (optionally signed) integer. y — maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* — additional units. The following units may be specified: *days* *byte* *hours* *2Kbyte* *minutes* *4Kbyte* *seconds* *Mbyte* *milliseconds*
x..y *special*	with the other data types: length specification The length specification is not shown for data types catid, date, device, product-version, time and vsn. x — minimum length for the operand value; x is an integer. y — maximum length for the operand value; y is an integer.. x=y — the length of the operand value must be precisely x *special* — additional specification for the description of a special data type which is checked by implementation. “special” may be preceded by other suffixes. The following entries are used: *arithm-expr* arithmetic expression (SDF-P) *bool-expr* logical expression (SDF-P) *string-expr* string expression (SDF-P) *expr* any expression (SDF-P) *cond-expr* conditional expression (JV)
with	Extends the specification options for a data type
-compl	When specifying the data type “date”, SDF expands two-digit year specifications in the form yy-mm-dd to: 20jj-mm-dd if yy < 60 19jj-mm-dd if yy ≥ 60
-low	Uppercase and lowercase letters are differentiated.
-path-compl	In the case of entries relating to the data type “filename”, SDF adds the catalog ID and/or user ID if they have not been specified.
-under	Permits underscores (_) for the data type “name”and “composed-name”.

<table>
<tr><th>Suffix</th><th colspan="2">Meaning</th></tr>
<tr><td>with
(continued)</td><td colspan="2"></td></tr>
<tr><td>-wild(n)</td><td colspan="2">Parts of names may be replaced by the following wildcards.
n denotes the maximum input length when using wildcards.
Due to the introduction of the data types posix-filename and posix-pathname, wildcards from the UNIX world (referred to below as POSIX wildcards) are now accepted in addition to the usual BS2000 wildcards. At present, not all commands support POSIX wildcards. As a result, their use with data types other than posix-filename and posix-pathname may lead to semantic errors. Only POSIX wildcards or only BS2000 wildcards should be used within any one search pattern. Only POSIX wildcards are allowed for the data types posix-filename and posix-pathname. If a pattern can be matched more than once in a string, the first match is used.</td></tr>
<tr><td></td><td>BS2000-
wildcard</td><td>Meaning</td></tr>
<tr><td></td><td>*</td><td>Replaces an arbitrary (even empty) character string. If the string concerned starts with *, then the * must be entered twice in succession if it is followed by other characters and if the character string entered does not contain at least one other wildcard.</td></tr>
<tr><td></td><td>Terminat-
ing period</td><td>Partially-qualified entry of a name.
Corresponds implicitly to the string “.*”, i.e. at least one other character follows the period.</td></tr>
<tr><td></td><td>/</td><td>Replaces any single character.</td></tr>
<tr><td></td><td><s_x:s_y></td><td>Replaces a string that meets the following conditions:
– It is at least as long as the shortest string (s_x or s_y)
– It is not longer than the longest string (s_x or s_y)
– It lies between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z, 0...9)
– s_x can also be an empty string (which is in the first position in the alphabetic collating sequence)
– s_y can also be an empty string, which in this position stands for the string with the highest possible code (contains only the characters X'FF')</td></tr>
</table>

Suffix	**Meaning**	
with-wild(n) (continued)	$<s_1,\ldots>$	Replaces all strings that match any of the character combinations specified by s. s may also be an empty string. Any such string may also be a range specification “s_x:s_y” (see above).
	-s	Replaces all strings that do not match the specified string s. The minus sign may only appear at the beginning of string s. Within the data type filename or partial-filename the negated string -s can be used exactly once, i.e. -s can replace one of the three name components: cat, user or file.
	Wildcards are not permitted in generation and version specifications for file names. Only system administration may use wildcards in user IDs. Wildcards cannot be used to replace the delimiters in name components cat (colon) and user ($ and period).	
	POSIX-wildcards	Meaning
	*	Replaces any single string (including an empty string). An * appearing at the first position must be duplicated if it is followed by other characters and if the entered string does not include at least one further wildcard.
	?	Replaces any single character; not permitted as the first character outside single quotes.
	[c_x-c_y]	Replaces any single character from the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters.
	[s]	Replaces exactly one character from string s. The expressions [c_x-c_y] and [s] can be combined into [s_1c_1-c_2s_2]
	[!c_x-c_y]	Replaces exactly one character not in the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters. The expressions [!c_x-c_y] and [!s] can be combined into [!s_1c_1-c_2s_2]
	[!s]	Replaces exactly one character not contained in string s. The expressions [!s] and [!c_x-c_y] can be combined into [!s_1c_1-c_2s_2]

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with
(continued)</td><td></td></tr>
<tr><td>-wild-
constr(n)</td><td>Specification of a constructor (string) that defines how new names are to be constructed from a previously specified selector (i.e. a selection string with wildcards). See also with-wild. n indicates the maximum input length when wildcards are used.
The constructor may consist of constant strings and patterns. A pattern (character) is replaced by the string that was selected by the corresponding pattern in the selector.
The following wildcards may be used in constructors:
<table>
<tr><th>Wildcard</th><th>Meaning</th></tr>
<tr><td>*</td><td>Corresponds to the string selected by the wildcard * in the selector.</td></tr>
<tr><td>Terminating period</td><td>Corresponds to the partially-qualified specification of a name in the selector;
corresponds to the string selected by the terminating period in the selector.</td></tr>
<tr><td>/ or ?</td><td>Corresponds to the character selected by the / or ? wildcard in the selector.</td></tr>
<tr><td><n></td><td>Corresponds to the string selected by the n-th wildcard in the selector, where n is an integer.</td></tr>
</table>
Allocation of wildcards to corresponding wildcards in the selector:
All wildcards in the selector are numbered from left to right in ascending order (global index).
Identical wildcards in the selector are additionally numbered from left to right in ascending order (wildcard-specific index).
Wildcards can be specified in the constructor by one of two mutually exclusive methods:
1. Wildcards can be specified via the global index: <n>
2. The same wildcard may be specified as in the selector; substitution is carried out on the basis of the wildcard-specific index:
For example, the second "/" corresponds to the string selected by the second "/" in the selector.</td></tr>
</table>

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with-wild-constr(n) (continued)</td><td>
The following rules must be observed when specifying a constructor:
– The constructor can only contain wildcards from the selector.
– If the string selected by the wildcard <...> or [...] is to be used in the constructor, the index notation must be selected.
– The index notation must be selected if the string identifying a wildcard in the selector is to be used more than once in the constructor:
For example, if "A/" is specified, the constructor "A<n><n>" must be specified instead of "A//".
– The wildcard * can also be a null (empty) string. Note that if two or more asterisks appear in sequence (even with further wildcards), only the last asterisk can be a non-null string, e.g. for “****” or “*//*”.
– Valid names must be produced by the constructor. This must be taken into account when specifying both the constructor and the selector.
– Depending on the constructor, identical names may be constructed from different names selected by the selector. For example:
“A/*” selects the names “A1” and “A2”; the constructor “B*” generates the same new name “B” in both cases.
To prevent this from occurring, all wildcards of the selector should be used at least once in the constructor.
– If the selector ends with a period, the constructor must also end with a period (and vice versa).

Examples:
<table>
<tr><td>Selector</td><td>Selection</td><td>Constructor</td><td>New name</td></tr>
<tr><td>A//*</td><td>AB1
AB2
A.B.C</td><td>D<3><2></td><td>D1
D2
D.CB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<3>.XY<2></td><td>G.A.D.XYA
G.A.D.XYB
G.B.F.XYA
G.B.F.XYB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<2>.XY<2></td><td>G.A.A.XYA
G.A.B.XYB
G.B.A.XYA
G.B.B.XYB</td></tr>
<tr><td>A//B</td><td>ACDB
ACEB
AC.B
A.CB</td><td>G/XY/</td><td>GCXYD
GCXYE
GCXY. [1]
G.XYC</td></tr>
</table>
[1] The period at the end of the name may violate naming conventions (e.g. for fully qualified file names)
</td></tr>
</table>

Suffix	**Meaning**
without	Restricts the specification options for a data type.
-cat	Specification of a catalog ID is not permitted
-corr	Input format: [[C]'][V][m]m.na['] Specifications for the data type product-version must not include the correction status.
-gen	Specification of a file generation or file generation group is not permitted.
-man	Input format: [[C]'][V][m]m.n['] Specifications for the data type product-version must not include either release or correction status.
-odd	The data type x-text permits only an even number of characters.
-sep	With the data type “text”, specification of the following separators is not permitted: ; = ( ) < > ? (i.e. semicolon, equals sign, left and right parentheses, greater than, less than, and blank).
-temp-file	Specification of a temporary file is not permitted (see #file or @file with filename).
-user	Specification of a user ID is not permitted.
-vers	Specification of the version (see “file(no)”) is not permitted for tape files.
-wild	The data types posix-filename and posix-pathname must not contain wildcards.
mandatory	Certain specifications are mandatory for a data type
-corr	Input format: [[C]'][V][m]m.naso['] Specifications for the data type product-version must contain the correction status (and thus also the release status).
-man	Input format: [[C]'][V][m]m.na[so]['] Specifications for the data type product-version must contain the release status. Specification of the correction status is possible unless explicitly prohibited by the suffix without-corr.
-quotes	Specifications for the data types posix-filename and posix-pathname must be enclosed in single quotes

Operators

Operator	Operation	Operator type	Operand (type)	Result value
+	Addition	Arithmetic operator	Integer	Integer
-	Subtraction	Arithmetic operator	Integer	Integer
*	Multiplication	Arithmetic operator	Integer	Integer
/	Division	Arithmetic operator	Integer	Integer
MOD	Modulo	Arithmetic operator	Integer	Integer
LT <	Less that	Relational operator	Integer, string	Boolean constant
LE <=	Less that or equal to	Relational operator	Integer, string	Boolean constant
EQ = ==	equal to	Relational operator	Integer, string	Boolean constant
NE <>	Not equal to	Relational operator	Integer, string	Boolean constant
GE >=	Greater than or equal to	Relational operator	Integer, string	Boolean constant
GT >	Greater than	Relational operator	Integer, string	Boolean constant
NOT	Negation	Logical operator	Logical expression	Boolean constant
OR	Or	Logical operator	Logical expression	Boolean constant
AND	And	Logical operator	Logical expression	Boolean constant
XOR	Exclusive or	Logical operator	Logical expression	Boolean constant
//	String concatenation	Concatenation operator	String expression	String

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *…@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *…@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.