SDF-P V2.4A

Programmieren in der Kommandosprache

Ausgabe Mai 2007

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@fujitsu-siemens.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2000 erfüllt.

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

Copyright und Handelsmarken

Inhalt

Einleitung

Dieses Tabellenheft dient zum schnellen Nachschlagen der Informationen, die für das Erstellen von S-Prozeduren benötigt werden. Ausführliche Informationen enthält das SDF-P Benutzerhandbuch.

Das Tabellenheft ist in fünf Abschnitte gegliedert:

– Übersicht und Kurzbeschreibung der SDF-P-Funktionen

– Übersicht, Beschreibung der Returncodes und Kurzbeschreibung der SDF-P-Kommandos

– Beschreibung der Assembler-Makroschnittstelle

– Beschreibung der Syntax für Variablennamen

– Beschreibung der Metasyntax, Datentypen und Operatoren

Vordefinierte Funktionen

Übersicht

Funktionen	Kurzbeschreibung
ACCOUNT( )	Liefert die Abrechnungsnummer der Task
ARRAY-INDEX( )	Liefert den Wert eines Arrayindex
BOOLEAN( )	Konvertiert einen Ausdruck in einen boolschen Wert
CHARACTER-TO-INTEGER( )	Konvertiert ein Zeichen (EBCDI-Code) in eine Integer-Zahl
CHECK-DATA-TYPE( )	Überprüft den SDF-Datentyp bei Strings bzw. Operandenwerten
COUNTER( )	Zählt die Aufrufe von COUNTER( )
CURRENT-TYPE( )	Gibt den aktuellen Typ des Werts einer einfachen Variablen an
DATE( )	Liefert das aktuelle Datum
DATE-VALUE( )	Gibt ein bestimmtes Datum aus
DAY( )	Liefert den Namen des aktuellen Wochentages
ELAPSED-DAYS( )	Gibt die Anzahl der Tage zwischen zwei Datumsangaben an
EXPLICIT-CALL( )	Informiert über die Art des Prozeduraufrufs
EXTEND-SDF-LIST( )	Fügt einer SDF-Liste ein Listenelement an
EXTRACT-FIELD( )	Trennt von einem Eingabe-String ein Feld ab
FILL( )	Füllt einen String mit Leerzeichen auf
FIRST-VARIABLE-NAME( )	Analysiert den Aufbau von zusammengesetzten Variablen
FROM-C-LITERAL( )	Konvertiert ein C-Literal in den entsprechenden Stringwert
FROM-X-LITERAL( )	Konvertiert ein X-Literal in seinen hexadezimalen Wert
HASH-STRING( )	Ausdruck als String verschlüsseln
HASH-VALUE( )	Ausdruck als Integer-Wert verschlüsseln
HOME-CAT-ID( )	Liefert die Katalogkennung des Home-Pubsets
HOST( )	Liefert den internen Namen des Rechners, auf dem die Prozedur läuft
INDEX( )	Zeigt die Position eines Suchstrings im Gesamtstring an
INSTALLATION-PATH( )	Installations-Pfadnamen einer Datei ausgeben
INTEGER( )	Konvertiert einen beliebigen Ausdruck in den Datentyp INTEGER
INTEGER-TO-CHARACTER( )	Konvertiert eine Zahl in ein Zeichen (C-String)
INTEGER-TO-X-LITERAL( )	Konvertiert eine Dezimalzahl in ein 4 Byte langes X-Literal, das die Codierung der Zahl enthält (inverse Funktion zu X-LITERAL-TO-INTEGER)

Funktionen	Kurzbeschreibung
IS-C-LITERAL( )	Prüft, ob der angegebene String ein C-Literal ist und in einen String konvertiert werden kann
IS-CATALOGED-FILE( )	Prüft, ob es einen Katalogeintrag mit dem angegebenen Dateinamen gibt
IS-CATALOGED-JV( )	Prüft, ob es einen Katalogeintrag für den angegebenen Jobvariablenamen gibt
IS-DECLARED( )	Prüft, ob die angegebene einfache oder zusammengesetzte Variable bereits deklariert ist
IS-EMPTY-FILE( )	Prüft, ob die Datei leer ist, wenn bei HIGHEST-USED-PAGES( ) der Wert 0 zurückgegeben wurde
IS-INITIALIZED( )	Prüft, ob die angegebene Variable initialisiert ist
IS-INTEGER( )	Prüft, ob der als String angegebene Ausdruck einen Integer-Wert darstellt
IS-LIBRARY( )	Prüft, ob die angegebene Datei im Katalog als PLAM-Bibliothek eingetragen ist
IS-LIBRARY-ELEMENT( )	Prüft, ob das angegebene Bibliothekselement existiert oder nicht
IS-SDF-LIST( )	Analysiert, ob ein String-Ausdruck eine SDF-Liste ist bzw. ein Kriterium für eine SDF-Liste erfüllt
IS-SDF-P( )	Prüft, ob SDF-P im System geladen ist
IS-SDF-STRUCTURE( )	Analysiert, ob der angegebene String eine SDF-Struktur ist
IS-VARIABLE-NAME( )	Prüft, ob der angegebene String ein syntaktisch korrekter Variablenname ist
IS-X-LITERAL( )	Prüft, ob der angegebene Stringausdruck ein X-Literal enthält und mit FROM-X-LITERAL( ) konvertiert werden kann
JOB-CLASS( )	Fragt ab, zu welcher Jobklasse die aktuelle Task gehört
JOB-MONJV( )	Liefert den Namen der Jobvariablen, die den Auftrag überwacht
JOB-NAME( )	Liefert den Jobnamen der aktuellen Task
JV( )	Liefert den Inhalt der angegebenen Jobvariablen
LAYOUT-SCOPE( )	Liefert den Geltungsbereich (Scope) für Strukturlayouts
LENGTH( )	Liefert die Länge des angegebenen Strings
LIMIT( )	Fragt ab, wie viele Elemente eine Listenvariable enthalten darf
LOGGING-MODE( )	Zeigt an, ob beim Prozeduraufruf Protokollierung von Kommandos oder Daten eingeschaltet wurde
LOWER-CASE( )	Setzt alle Großbuchstaben im angegebenen String in Kleinbuchstaben um
MAINCODE( )	Liefert den sieben Byte langen Fehlerschlüssel des Returncodes
MONTH( )	Liefert den aktuellen Monatsnamen in der angegebenen Sprache

Funktionen	Kurzbeschreibung
MSG( )	Liefert den Meldungstext, der dem angegebenen Meldungsschlüssel zugeordnet ist
NEXT-VARIABLE-NAME( )	Analysiert den Aufbau von zusammengesetzten Variablen
PROC-LEVEL( )	Liefert die aktuelle Schachtelungstiefe der S-Prozedur
PROCESSOR( )	Liefert den Prozessornamen der TIAM-Station
PROG-MONJV( )	Liefert den Namen der Jobvariablen
PROG-NAME( )	Liefert den auf acht Zeichen abgeschnittenen internen Namen des aktuell geladenen Programms
RENAME( )	Liefert einen neuen Namen
REPLACE( )	Ersetzt einen Teilstring innerhalb eines Strings durch einen anderen String
RUN-PRIORITY( )	Liefert die Prioritätsstufe des aktuellen Jobs
SDF-P-VERSION( )	aktuelle Version von SDF-P bzw. SDF-P-BASYS abfragen
SDF-STRUCTURE-VALUE( )	Liefert den Inhalt einer SDF-Struktur teilweise oder ganz
SEARCH-LIST-INDEX( )	String in einer Liste suchen
SESSION-NUMBER( )	Liefert die Systemlaufnummer des aktuell laufenden Systems
SIZE( )	Fragt ab, aus wie vielen Elementen die angegebene Variable besteht
STATION( )	Liefert den Stationsnamen der TIAM-Station, von der die Prozedur angestoßen wurde
STATION-TYPE( )	Liefert den generierten Gerätetyp der TIAM-Station, von der die Prozedur aufgerufen wurde
STD-CAT-ID( )	Liefert die Kennung des Pubsets, der der aktuellen Benutzerkennung als Default-Pubset zugeteilt wurde
STMT-SPINOFF( )	Zeigt an, ob für das geladene Programm ein Statement-Spinoff eingeschaltet ist
STRING( )	Konvertiert einen Ausdruck vom Typ INTEGER, BOOLEAN oder STRING in den Typ STRING
SUBCODE1( )	Liefert Subcode1(Fehlerklasse) des aktuellen Kommando-Returncodes
SUBCODE2( )	Liefert Subcode2 (Zusatzinformation) des aktuellen Kommando-Returncodes
SUBLIST( )	Liefert den Inhalt des gewählten Elements einer SDF-Liste
SUBLIST-NUMBER( )	Gibt aus, wie viele Elemente eine SDF-Liste enthält
SUBSTRING( )	Extrahiert einen Teilstring aus dem angegebenen String
SYSCMD( )	Liefert den Namen der Datei, die der Systemdatei SYSCMD zugewiesen ist

Funktionen	Kurzbeschreibung
SYSDTA( )	Liefert den Namen der Datei, die der Systemdatei SYSDTA zugewiesen ist
SYS-ID( )	Liefert das Systemkennzeichen (System-ID)
SYSLST( )	Liefert den Namen der Datei, die der Systemdatei SYSLST zugewiesen ist
SYSOUT( )	Liefert den Namen der Datei, die der Systemdatei SYSOUT zugewiesen ist
SYSTEM-CALL( )	Informiert, in welcher Syntaxdatei das implementierte Kommando steht
SYSTEM-INFORMATION( )	Informiert über Systemparameter und Klasse-2-Optionen
TASK-MODE( )	Liefert den Modus der aktuellen Task
TIME( )	Liefert die aktuelle Uhrzeit
TO-C-LITERAL( )	Konvertiert den angegebenen String in ein C-Literal
TO-X-LITERAL( )	Konvertiert den Hexadezimalwert eines Strings in die externe Darstellung
TRANSLATE( )	Ordnet beliebigen Eingangswerten beliebige Ergebniswerte zu
TRANSLATE-BOOLEAN( )	Prüft, ob der Eingabe-Ausdruck wahr oder falsch ist
TRIM( )	Entfernt gleiche Zeichen am Anfang oder Ende
TSN( )	Liefert die Auftragsnummer des aktuellen Auftrags
UPPER-CASE( )	Setzt alle Kleinbuchstaben im angegebenen String in Großbuchstaben um
USER-IDENTIFICATION( )	Liefert die Benutzerkennung des aktuellen Auftrags
USER-SWITCH( )	Prüft den Wert des angegebenen Benutzerschalters
VARIABLE-ATTRIBUTE( )	Liefert für die angegebene Variable den Wert des angegebenen Attributs
VARIABLE-TO-STRING( )	Konvertiert eine S-Variable vom Typ STRUKTUR in einen String
VERIFY( )	Prüft zwei Strings und liefert als Ergebnis die Position des ersten Zeichens im String (STRING), das nicht im Suchstring (PATTERN) enthalten ist
WILDCARD( )	Vergleicht einen String mit einem Musterstring
X-LITERAL-TO-INTEGER( )	Konvertiert einen maximal 4 Byte langen String in eine Dezimalzahl (inverse Funktion zu INTEGER-TO-X-LITERAL)

Funktionen

ACCOUNT( ) Abrechnungsnummer abfragen

Ermittelt die Abrechnungsnummer der aktuellen Task, die im SET-LOGON-PARAMETERS-Kommando angegeben wurde.

Ergebnistyp: STRING (<string 1 .. 8>)

ACCOUNT( )

ARRAY-INDEX( ) Arrayindex abfragen

Die Funktion ARRAY-INDEX( ) lässt sich auf Arrays anwenden. ARRAY-INDEX liefert den Wert eines Arrayindex. Dadurch können andere Funktionen anschließend über den Arrayindex explizit auf dieses Element zugreifen.

Ergebnistyp: INTEGER

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

BOOLEAN( ) Konvertieren in booleschen Wert

Konvertiert den Ausdruck, der beim Funktionsaufruf angegeben wird, in einen Wert vom Typ BOOLEAN.

Ergebnistyp: BOOLEAN

BOOLEAN( ) BOOLE( )
EXPRESSION = ausdruck

CHARACTER-TO-INTEGER( ) Zeichen in Zahl konvertieren

Konvertiert *ein* Zeichen in eine Dezimalzahl, ausgehend von der Codierung des Zeichens im EBCDI-Code .

Besteht der Eingabestring aus mehreren Zeichen, wird nur das erste Zeichen umgesetzt.

Durch Kombination mit den entsprechenden String-Funktionen (z.B. SUBSTRING) können aber auch alle Zeichen eines Strings konvertiert werden.

Ergebnistyp: INTEGER (<integer 0..255>)

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

CHECK-DATA-TYPE( ) Operandenwert überprüfen

Prüft bei Strings bzw. Operandenwerten, ob sie SDF-Datentyp-Bestimmungen erfüllen.

Ergebnistyp: BOOLEAN

DATA-TYPE=	**erlaubte Operandenkombinationen**
*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_ausdruck
,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_ausdruck
,LONGEST-LENGTH = *ANY / arithm_ausdruck
,LONGEST-LOGICAL-LENGTH = *NONE / arithm_ausdruck
,DECIMAL-DIGITS-SHORTEST = 0 / arithm_ausdruck
,DECIMAL-DIGITS-LONGEST = 0 / arithm_ausdruck
,VALUE = *NO / list-poss: string_ausdruck
,PATTERN = *NO / string_ausdruck
,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_ausdruck
,EXCEPT-TAPES = *NONE / list-poss: string_ausdruck

COUNTER( ) Funktionsaufrufe zählen

Ermittelt die Anzahl der Aufrufe von COUNTER( ) in der aktuellen Task. Nach jedem Aufruf von COUNTER( ) wird der Zähler um 1 erhöht.

Ergebnistyp: INTEGER (<integer 1 .. 2147483647>)

COUNTER( )

CURRENT-TYPE( ) Variablentyp abfragen

Gibt den aktuellen Typ des Werts einer einfachen Variable zurück (dieser darf nicht mit dem aktuellen Typ einer Variablendeklaration verwechselt werden, der wiederum mit VARIABLE-ATTRIBUTE( ) zurückgegeben wird). Wenn der Variablentyp noch nicht festgelegt ist (TYPE = *ANY) oder CURRENT-TYPE( ) auf eine zusammengesetzte Variable angewendet wird, wird als Ergebnis *NONE zurückgegeben.

Ergebnistyp: STRING

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

DATE( ) Datum ausgeben

Ermittelt das aktuelle Datum und gibt es im gewählten Format zurück.

Ergebnistyp: STRING (<string 10..13>)

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

DATE-VALUE( ) Bestimmtes Datum ausgeben

Gibt ein bestimmtes Datum aus, das einer angegebenen Zahl von Tagen seit dem Basisdatum entspricht (Defaultwert ist der Beginn des 20. Jahrhunderts, d.h. das Datum 1900-01-01).

Ergebnistyp: STRING (<string 10..13>

DATE-VALUE( )
NUMBER-OF-DAYS = arithm_ausdruck ,BASE = *STD / *TODAY / string_ausdruck ,FORMAT = *ISO / *AMERICAN / *GERMAN

DAY( ) Wochentag ausgeben

Liefert den Namen des aktuellen Wochentages in der angegebenen Sprache, allerdings nur in einer Kurzform.

Ergebnistyp: STRING (<string 2..3>)

DAY( )
LANGUAGE = *ENGLISH / *GERMAN / *STD

ELAPSED-DAYS( ) Differenztage ausgeben

Gibt die Anzahl der Differenztage zwischen zwei Datumsangaben an.

Ergebnistyp: INTEGER (<integer -3074323 .. 3074323>)

ELAPSED-DAYS( )
DATE = string_ausdruck ,BASE = *STD / *TODAY / string_ausdruck

EXPLICIT-CALL( ) Expliziten Prozeduraufruf ausgeben

Gibt die Art des Prozeduraufrufs aus: Dabei bedeutet TRUE einen expliziten Aufruf (d.h. mit CALL-PROCEDURE, INCLUDE-PROCEDURE) und FALSE einen impliziten Aufruf (z.B. wenn der Aufruf von einem Kommando einer Prozedur erfolgte.

Ergebnistyp: BOOLEAN

EXPLICIT-CALL( )

EXTEND-SDF-LIST( ) Listenelement anfügen

Fügt bei einer SDF-Liste ein neues Element hinzu. Dieses neue Element kann wiederum eine SDF-Liste sein.

Ergebnistyp: STRING

EXTEND-SDF-LIST( )
LIST = string_ausdruck ,ELEMENT = string_ausdruck ,POSITION = *LAST / *FIRST / arithm_ausdruck

EXTRACT-FIELD( ) Feld abtrennen

Trennt von einem Eingabe-String ein Feld ab.

Ergebnistyp: STRING

EXTRACT-FIELD( )
STRING = string_ausdruck ,FIELD-NUMBER = arithm_ausdruck ,FIELD-SEPARATOR = *ANY-BLANKS / string_ausdruck

FILL( ) String auffüllen

Füllt den String, der im Funktionsaufruf angegeben wurde, bis zur angegebenen Länge und in der angegebenen Richtung mit Füllzeichen auf. Diese Füllzeichen können im Funktionsaufruf definiert werden.

Ergebnistyp: STRING

FILL( )
STRING = string_ausdruck ,LENGTH = arithm_ausdruck ,SIDE = *RIGHT / *LEFT ,FILL-BYTE = C'␣' / zeichen

FIRST-VARIABLE-NAME( ) Variablenelementnamen abfragen

Mit der Funktion FIRST-VARIABLE-NAME( ) lässt sich der Aufbau von zusammengesetzten Variablen analysieren, vor allem in Kombination mit der Funktion NEXT-VARIABLE-NAME( ). Die Funktion FIRST-VARIABLE-NAME( ) kann auf alle Aggregate und Listen angewendet werden. Dabei geht FIRST-VARIABLE-NAME( ) vom angegebenen Variablen- oder Variablenelementnamen aus und liefert dann den Namen der ersten Variablen. Gibt es keine tiefere Ebene, liefert FIRST-VARIABLE-NAME( ) das Ergebnis *END.

Ergebnistyp: STRING (<composed-name 1..255> oder '*END')

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

FROM-C-LITERAL( ) C-Literal konvertieren

Konvertiert ein C-Literal in den entsprechenden Stringwert. Dabei werden das vorangestellte C und die Hochkommas am Anfang und Ende des Literals gelöscht.

FROM-C-LITERAL( ) ist die inverse Funktion zu TO-C-LITERAL( ).

Ergebnistyp: STRING

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

FROM-X-LITERAL( ) X-Literal konvertieren

Konvertiert ein X-Literal in den entsprechenden Stringwert.

FROM-X-LITERAL( ) ist die inverse Funktion zu TO-X-LITERAL( ).

Ergebnistyp: STRING

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

HASH-STRING( ) Ausdruck als String verschlüsseln

Konvertiert einen String-Ausdruck in einen String binären Inhalts mit frei zu wählender Länge (z.B. ein Passwort). Der verwendete Algorithmus liefert zu unterschiedlichen Strings mit hoher Wahrscheinlichkeit unterschiedliche Ausgaben.

Ergebnistyp: STRING

HASH-STRING( )
STRING = string_ausdruck ,LENGTH = 4 / arithm_ausdruck

HASH-VALUE( ) Ausdruck als Integer-Wert verschlüsseln

Konvertiert einen String-Ausdruck in einen Integer-Wert, wobei der verwendete Algorithmus zu unterschiedlichen Strings mit hoher Wahrscheinlichkeit unterschiedliche Ausgabewerte erzeugt.

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

HASH-VALUE( )
STRING = string_ausdruck

HOME-CAT-ID( ) Katalogkennung des Home-Pubsets abfragen

Liefert die Katalogkennung des Home-Pubsets.
Die Katalogkennung eines Pubsets wird von der Systembetreuung beim Einrichten des Pubsets vergeben und kann bis zu vier Zeichen lang sein. Der Home-Pubset des laufenden Systems ist der Pubset, von dem das System geladen wurde.

Ergebnistyp: STRING (<string 1..4>)

HOME-CAT-ID( )

HOST( ) Rechnernamen abfragen

Liefert den internen Namen des Rechners, auf dem die Funktion aufgerufen wird. Den internen Rechnernamen legt die Systembetreuung beim Starten des DCM fest.

Ergebnistyp: STRING (<string 1..8>)

HOST( )

INDEX( ) String suchen

Zeigt die Position eines Suchstrings im Gesamtstring an. Der Gesamtstring kann von links nach rechts oder von rechts nach links durchsucht werden; der Ergebniswert bezieht sich immer auf den Anfang des Gesamtstrings.

Ergebnistyp: INTEGER

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

INSTALLATION-PATH( ) Pfadnamen ausgeben

Gibt für den logischen Namen einer Datei (Installation-Item), die zu einer bestimmten Produktversion gehört, den zugeordneten Pfadnamen aus dem SCI aus.

Ergebnistyp: STRING

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

INTEGER( ) Ausdruck in Integer-Wert konvertieren

Konvertiert einen beliebigen Ausdruck in den Datentyp INTEGER.

Ergebnistyp: INTEGER

INTEGER( ) INT( )
EXPRESSION = ausdruck

INTEGER-TO-CHARACTER( ) Zahl in Zeichen konvertieren

Konvertiert eine Zahl in ein Zeichen(C-String).

Die Zahl wird als Integer-Wert des EBCDI-Codes eines Zeichens interpretiert. Das so codierte Zeichen wird als Ergebnis zurückgegeben.

Die Funktion INTEGER-TO-CHARACTER( ) ist die Umkehrung der Funktion CHARACTER-TO-INTEGER( ).

Ergebnistyp: STRING (<string 1..1>)

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

INTEGER-TO-X-LITERAL( ) Zahl in X-Literal konvertieren

Konvertiert eine Dezimalzahl in ein X-Literal, das die 4 Byte lange Codierung der Zahl enthält.

INTEGER-TO-X-LITERAL( ) ist die inverse Funktion zu X-LITERAL-TO-INTEGER( ).

Ergebnistyp: STRING

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

IS-C-LITERAL( ) C-Literal prüfen

Prüft, ob der angegebene String ein C-Literal ist und in einen String konvertiert werden kann (mit der Funktion FROM-C-LITERAL).

Ergebnistyp: BOOLEAN

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

IS-CATALOGED-FILE( ) Katalogeintrag prüfen

Die Funktion IS-CATALOGED-FILE( ) prüft, ob es einen Katalogeintrag mit dem angegebenen Dateinamen gibt. Die Reaktion im Fehlerfall (ungültiger Dateiname usw.) kann vereinbart werden.

Ergebnistyp: BOOLEAN

IS-CATALOGED-FILE( ) IS-CAT-FILE( )
FILE = string_ausdruck ,ERROR-REPORTING = *PROC-ERROR-MECHANISM / *RETURN-FALSE ,ERROR-VARIABLE = *NONE / string_ausdruck

IS-CATALOGED-JV( ) Jobvariable abfragen

Die Funktion IS-CATALOGED-JV( ) prüft, ob es einen Katalogeintrag für den angegebenen Jobvariablennamen gibt, das heißt, ob die angegebene Jobvariable existiert.

Ergebnistyp: BOOLEAN

IS-CATALOGED-JV( ) IS-CAT-JV( )
JV = string_ausdruck ,ERROR-REPORTING = *PROC-ERROR-MECHANISM / *RETURN-FALSE ,ERROR-VARIABLE = *NONE / string_ausdruck

IS-DECLARED( ) Variablendeklaration prüfen

Prüft, ob die angegebene einfache oder zusammengesetzte Variable bereits deklariert ist.

Ergebnistyp: BOOLEAN

IS-DECLARED( )
VARIABLE-NAME = string_ausdruck ,SCOPE = *BY-HIERARCHY / *TASK / *CALLING-PROCEDURES

IS-EMPTY-FILE( ) Dateigröße prüfen

Prüft, ob die Datei leer ist (Last-Page-Pointer zeigt auf die Seite 0). Das ist der Fall, wenn in der Ausgabe des Kommandos SHOW-FILE-ATTRIBUTES das Ausgabefeld HIGH-US-PA den Wert 0 enthält.

Ergebnistyp: BOOLEAN

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

IS-INITIALIZED( ) Variableninitialisierung prüfen

Prüft, ob die angegebene Variable initialisiert ist, das heißt, ob sie einen gültigen Inhalt hat. Auch der Leerstring ist ein gültiger Variableninhalt.

Eine Variable kann nur dann initialisiert werden, wenn sie deklariert ist.
Es können nur einfache Variablen oder Listenvariablen geprüft werden.

Ergebnistyp: BOOLEAN

IS-INITIALIZED( )
VARIABLE-NAME = string_ausdruck

IS-INTEGER( ) Ausdruck prüfen

Prüft, ob der als String angegebene Ausdruck einen Integer-Wert darstellt.

Ergebnistyp: BOOLEAN

IS-INTEGER( )
STRING = string_ausdruck

IS-LIBRARY( ) Bibliotheksname prüfen

Prüft, ob die angegebene Datei im Katalog als PLAM-Bibliothek eingetragen ist.

Wenn die Datei im Katalog nicht als PLAM-Bibliothek eingetragen ist, liefert IS-LIBRARY( ) das Ergebnis FALSE.

Ergebnistyp: BOOLEAN

IS-LIBRARY( )
FILE = string_ausdruck

IS-LIBRARY-ELEMENT( ) Bibliothekselement prüfen

Prüft, ob das angegebene Bibliothekselement existiert oder nicht.

Ergebnistyp: BOOLEAN

IS-LIBRARY-ELEMENT( ) IS-LIB-ELEM( )
LIBRARY = string_ausdruck ,ELEMENT = string_ausdruck ,TYPE = string_ausdruck ,VERSION = *HIGHEST-EXISTING / string_ausdruck

IS-SDF-LIST( ) String auf Kriterien für SDF-Liste analysieren

Analysiert, ob ein String-Ausdruck eine SDF-Liste der Form '(<element$_1$>,<element$_2$>,..., <element$_n$>)' ist . Dabei sind <element$_i$> Folgen von Zeichen, die kein Komma (außerhalb von Klammerpaaren) enthalten.

Ergebnistyp: BOOLEAN

IS-SDF-LIST( )
STRING = string_ausdruck

IS-SDF-P( ) Prüfen, ob SDF-P geladen ist

Prüft, ob SDF-P im System geladen ist. Wenn es geladen ist, wird als Ergebnis TRUE zurückgegeben.

Ergebnistyp: BOOLEAN

IS-SDF-P( )

IS-SDF-STRUCTURE( ) String auf Kriterien für SDF-Struktur analysieren

Analysiert, ob der angegebene String eine SDF-Struktur ist.

Die angegebene SDF-Struktur muss mit einem Wert eingeleitet werden. Dieser Wert kann nicht weggelassen werden, ansonsten wird der String als SDF-Liste und nicht als SDF-Struktur betrachtet.

Ergebnistyp: BOOLEAN

IS-SDF-STRUCTURE( )
STRING = string_ausdruck

IS-VARIABLE-NAME( ) Variablennamen prüfen

Prüft, ob der angegebene String ein syntaktisch korrekter Variablenname ist. Diese Prüfung ist eine reine Syntaxprüfung. Es wird nicht geprüft, ob es eine Variable dieses Namens gibt.

Ergebnistyp: BOOLEAN

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

IS-X-LITERAL( ) X-Literal prüfen

Prüft, ob der angegebene Stringausdruck ein X-Literal enthält und mit FROM-X-LITERAL( ) konvertiert werden kann.

Ergebnistyp: BOOLEAN

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

JOB-CLASS( ) Jobklasse abfragen

Fragt ab, zu welcher Jobklasse die aktuelle Task gehört.

Ergebnistyp: STRING

JOB-CLASS( )

JOB-MONJV( ) MONJV abfragen

Liefert den Namen der Jobvariablen, die den Auftrag überwacht.

Ergebnistyp: STRING

JOB-MONJV( )

JOB-NAME( ) Jobnamen abfragen

Liefert den Jobnamen der aktuellen Task, das heißt, den Namen, der im Kommando SET-LOGON-PARAMETERS angegeben wurde.

Ergebnistyp: STRING (<string 1..8>)

JOB-NAME( )

JV( ) Jobvariable abfragen

Liefert den Inhalt der angegebenen Jobvariablen.

Ergebnistyp: STRING

JV( )
JV-NAME = string_ausdruck ,START = 1 / arithm_ausdruck$_1$,LENGTH = *REST-LENGTH / arithm_ausdruck$_2$

LAYOUT-SCOPE( ) Layout-Geltungsbereich abfragen

Gilt nur für Strukturlayouts und liefert deren Geltungsbereich (Scope).

Ergebnistyp: STRING

LAYOUT-SCOPE( )
LAYOUT-NAME = string_ausdruck

LENGTH( ) Stringlänge abfragen

Liefert die Länge des angegebenen Strings.

Ergebnistyp: INTEGER

LENGTH( )
STRING = string_ausdruck

LIMIT( ) Maximale Listengröße abfragen

Die Funktion LIMIT( ) ist nur auf Listenvariable anwendbar. LIMIT( ) fragt ab, wie viele Elemente eine Listenvariable enthalten darf.

Ergebnistyp: INTEGER (<integer 1 .. 2147483647>)

LIMIT( )
LIST-NAME = string_ausdruck

LOGGING-MODE( ) Protokollierung prüfen

Zeigt an, ob beim Aufruf des Kommandos CALL-PROCEDURE oder INCLUDE-PROCEDURE Protokollieren eingeschaltet wurde.

Ergebnistyp: STRING ('YES' / 'NO')

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

LOWER-CASE( ) Großbuchstaben in Kleinbuchstaben umsetzen

Setzt alle Großbuchstaben im angegebenen String in Kleinbuchstaben um.

Die umzusetzenden Buchstaben müssen dem Standard-EBCDI-Code entsprechen.

Ergebnistyp: STRING

LOWER-CASE( )
STRING =string_ausdruck ,TRANSLATE = *ALL / *OUTSIDE-QUOTES-ONLY / *INSIDE-QUOTES-ONLY

MAINCODE( ) Fehlerschlüssel abfragen

Greift auf den Returncode des letzten Kommandos zu, das einen Fehler auslöste oder dem ein /SAVE-RETURNCODE folgte. Sie liefert den sieben Byte langen Fehlerschlüssel des Returncodes, der gleichzeitig der Meldungsschlüssel für eine Fehlermeldung ist (die übrigen Komponenten des Kommando-Returncodes werden mit den Funktionen SUBCODE1( ) und SUBCODE2( ) abgefragt).

Ergebnistyp: STRING (<string 7..7>)

MAINCODE( ) MC( )

MONTH( ) Monatsnamen ausgeben

Liefert den aktuellen Monatsnamen in der angegebenen Sprache, und zwar als drei Zeichen langer Kurzname.

Ergebnistyp: STRING (<string 1..3>)

MONTH( )
LANGUAGE = *ENGLISH / *GERMAN / *STD

MSG( ) Meldungstext ausgeben

Liefert den Meldungstext, der dem angegebenen Meldungsschlüssel zugeordnet ist, und zwar in der angegebenen Sprache und mit dem angegebenen Ausgabeformat.

Ergebnistyp: STRING (<string>)

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

NEXT-VARIABLE-NAME( ) Variablenebene abfragen

Mit der Funktion NEXT-VARIABLE-NAME( ) lässt sich der Aufbau von zusammengesetzten Variablen analysieren, vor allem in Verbindung mit der Funktion FIRST-VARIABLE-NAME( ).

Die Funktion NEXT-VARIABLE-NAME( ) liefert den Namen des in der gleichen Ebene folgenden Variablenelements. Wenn es kein Folgeelement auf dieser Ebene mehr gibt, liefert NEXT-VARIABLE-NAME( ) als Ergebnis *END.

Ergebnistyp: STRING

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

PROC-LEVEL( ) Schachtelungstiefe abfragen

Liefert die aktuelle Schachtelungstiefe der S-Prozedur.

Ergebnistyp: INTEGER

PROC-LEVEL( )

PROCESSOR( ) Prozessornamen abfragen

Liefert den Prozessornamen der TIAM-Station, das heißt der Datenstation, an der der aktuelle Auftrag gestartet wurde.

Ergebnistyp: STRING (<string 1..8>)

PROCESSOR( )

PROG-MONJV( ) Programm-MONJV abfragen

Liefert den Namen der Jobvariablen, die das geladene Programm überwacht.

Ergebnistyp: STRING (<string 1..255>)

PROG-MONJV( )

PROG-NAME( ) Programmnamen abfragen

Liefert den auf acht Zeichen abgeschnittenen internen Namen des aktuell geladenen Programms.

Ergebnistyp: STRING

PROG-NAME( )

RENAME( ) Neue Namen mit Wildcards bilden

Liefert einen neuen Namen. Der neue Name wird auf der Basis des Eingabenamens gebildet mithilfe von Wildcards.

Ergebnistyp: STRING

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

REPLACE( ) Teilstring überschreiben oder ersetzen

Überschreibt oder ersetzt einen Teilstring innerhalb eines Strings durch einen anderen String. Dabei kann der ursprüngliche String länger werden.

Ergebnistyp: STRING

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

RUN-PRIORITY( ) Laufzeitpriorität abfragen

Liefert die Prioritätsstufe des aktuellen Jobs.

Der gelieferte Wert kann anschließend geprüft und dann - wenn nötig - die Priorität der Task verändert werden.

Ergebnistyp: INTEGER (<integer 0..255>)

RUN-PRIORITY( ) RUN-PRIO( )

SDF-P-VERSION( ) SDF-P-Version abfragen

Informiert über die installierte Version des (kostenpflichtigen) Subsystems SDF-P bzw. über die aktuelle Version des im Grundausbau enthaltenen Subsystems SDF-P-BASYS.

Ergebnistyp: STRING

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

SDF-STRUCTURE-VALUE( ) Wert einer Struktur ausgeben

Liefert den Inhalt einer SDF-Struktur teilweise oder ganz.

Ergebnistyp: STRING (<string>)

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

SEARCH-LIST-INDEX( ) String in einer Liste suchen

Sucht in einer Listenvariablen einen String oder einen nach POSIX-Regeln gebildeten regulären Ausdruck. Der Returnwert gibt die Nummer des ersten Listenelements an, das diesen Ausdruck bzw. diesen Suchstring enthält.

Ergebnistyp: INTEGER

SEARCH-LIST-INDEX( )
LIST-VARIABLE-NAME = $string\_ausdruck_1$,PATTERN = $string\_ausdruck_2$,BEGIN-INDEX = 1 / arithm_ausdruck ,END-INDEX = *LAST / arithm_ausdruck ,BEGIN-COLUMN = 1 / arithm_ausdruck ,END-COLUMN = *LAST / arithm_ausdruck ,PATTERN-TYPE = *STRING / *REGULAR-EXPRESSION ,DIRECTION = *FORWARD / *REVERSE

SESSION-NUMBER( ) Systemlaufnummer abfragen

Liefert die Systemlaufnummer des aktuell laufenden Systems (die Systemlaufnummer ist z.B. Bestandteil des CONSLOG-Dateinamens).

Ergebnistyp: STRING (<string 3..3>)

SESSION-NUMBER( )

SIZE( ) Größe zusammengesetzter Variablen abfragen

Fragt ab, aus wie vielen Elementen die angegebene Variable besteht.

Ergebnistyp: INTEGER

SIZE( )
VARIABLE-NAME = string_ausdruck

STATION( ) TIAM-Stationsnamen abfragen

Liefert den Stationsnamen der TIAM-Station, von der die Prozedur angestoßen wurde.

Ergebnistyp: STRING(<string 1..8>)

STATION( )

STATION-TYPE( ) TIAM-Gerätetyp abfragen

Liefert den generierten Gerätetyp der TIAM-Station, von der die Prozedur aufgerufen wurde.

Ergebnistyp: STRING(<string 1..8>)

STATION-TYPE( )

STD-CAT-ID( ) Katalogkennung abfragen

Liefert die Kennung des Pubsets, der der aktuellen Benutzerkennung als Default-Pubset zugeteilt wurde.

Ergebnistyp: STRING (<string 1..4>)

STD-CAT-ID( )

STMT-SPINOFF( ) Statement-Spinoff abfragen

Zeigt an, ob für das geladene Programm ein Statement-Spinoff eingeschaltet ist.

Ergebnistyp: STRING (YES / NO / UNDEFINED)

STMT-SPINOFF( )

STRING( ) Ausdruck in String konvertieren

Konvertiert einen Ausdruck vom Typ INTEGER, BOOLEAN oder STRING in den Typ STRING. Es gelten die Regeln für implizites Konvertieren.

Ergebnistyp: STRING

STRING( ) STR( )
EXPRESSION = ausdruck

SUBCODE1( ) Subcode1 abfragen

Liefert die Fehlerklasse des aktuellen Kommando-Returncodes, das heißt des Returncodes des Kommandos, das einen Fehler auslöste oder dem das Kommando SAVE-RETURNCODE folgte.

Ergebnistyp: INTEGER (<integer 0..255>)

SUBCODE1( ) SC1( )

SUBCODE2( ) Subcode2 abfragen

Liefert die Gewichtung des aktuellen Kommando-Returncodes, das heißt des Returncodes des letzten Kommandos, das einen Fehler auslöste oder dem das Kommando SAVE-RETURNCODE folgte.

Ergebnistyp: INTEGER (<integer 0..255>)

SUBCODE2( ) SC2( )

SUBLIST( ) Element aus einer SDF-Liste wählen

Liefert den Inhalt des gewählten Elements einer SDF-Liste. Eine SDF-Liste ist ein String, der nach den syntaktischen Regeln für Operandenlisten in Kommandos interpretiert wird. Die Auswertung des Strings ist nur dann mit dieser Funktion sinnvoll durchzuführen, wenn er die Form '(<element$_1$>,<element$_2$>,...,<element$_n$>)' aufweist. Dabei sind <element$_i$> Folgen von Zeichen, die kein Komma (außerhalb von Klammerpaaren) enthalten.
Ob eine SDF-Liste vorliegt, kann mit der Funktion IS-SDF-LIST() geprüft werden.

Ergebnistyp: STRING

SUBLIST( )
LIST = string_ausdruck ,POSITION = arithm_ausdruck

SUBLIST-NUMBER( ) Elementanzahl einer SDF-Liste abfragen

Gibt aus, wie viele Elemente eine SDF-Liste enthält. Eine SDF-Liste ist ein String, der nach den syntaktischen Regeln für Operandenlisten in Kommandos interpretiert wird. Die Auswertung des Strings ist nur dann mit dieser Funktion sinnvoll durchzuführen, wenn er die Form '(<element$_1$>,<element$_2$>,...,<element$_n$>)' aufweist. Dabei sind <element$_i$> Folgen von Zeichen, die kein Komma (außerhalb von Klammerpaaren) enthalten.
Ob eine SDF-Liste vorliegt, kann mit der Funktion IS-SDF-LIST( ) geprüft werden.

Ergebnistyp: INTEGER

SUBLIST-NUMBER( )
LIST = string_ausdruck

SUBSTRING( ) Teilstring ausgeben

Extrahiert einen Teilstring aus dem angegebenen String. Über die Eingabeparameter werden die Anfangsposition des Teilstrings und seine Länge bestimmt.

Ergebnistyp: STRING

SUBSTRING( ) SUBSTR( )
STRING = string_ausdruck ,START = 1 / arithm_ausdruck$_1$,LENGTH = *REST-LENGTH / arithm_ausdruck$_2$

SYSCMD( ) SYSCMD-Zuweisung abfragen

Liefert den Namen der Datei (alternativ ist auch ein Bibliothekselement oder eine Listenvariable möglich), die der Systemdatei SYSCMD zugewiesen ist.

Ergebnistyp: STRING

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

SYSDTA( ) SYSDTA-Zuweisung abfragen

Liefert den Namen der Datei (alternativ ist auch ein Bibliothekselement oder eine Listenvariable möglich), die der Systemdatei SYSDTA zugewiesen ist.

Ergebnistyp: STRING

SYSDTA( )

SYS-ID( ) Systemkennzeichen abfragen

Liefert das Systemkennzeichen, das heißt die System-ID des laufenden Systems.

Ergebnistyp: STRING (<string 1..4>)

SYS-ID( )

SYSLST( ) SYSLST-Zuweisung abfragen

Liefert den Namen der Datei (alternativ ist auch ein Bibliothekselement oder eine Listenvariable möglich), die der Systemdatei SYSLST zugewiesen ist.

Ergebnistyp: STRING

SYSLST( )

SYSOUT( ) SYSOUT-Zuweisung abfragen

Liefert den Namen der Datei (alternativ ist auch ein Bibliothekselement oder eine Listenvariable möglich), die der Systemdatei SYSOUT zugewiesen ist

Ergebnistyp: STRING

.

SYSOUT( )

SYSTEM-CALL( ) Kommandoquelle ausgeben

Gibt innerhalb einer Prozedur an, die als Implementor eines Kommandos aufgerufen wurde, ob das Kommando aus der System- oder Gruppensyntaxdatei stammt.

Ergebnistyp: BOOLEAN

SYSTEM-CALL( )

SYSTEM-INFORMATION( ) Systemwerte abfragen

Mit der Funktion SYSTEM-INFORMATION( ) können Systeminformationen und Systemparameter abgefragt werden. Pro Aufruf kann ein Wert abgefragt werden.

Einschränkung

Die Funktion entspricht auf Programmebene dem Makroaufruf SINF, der nur noch kompatibel unterstützt wird und durch die Makroaufrufe NSIINF und NSIOPT ersetzt wurde. Systeminformationen bzw. Systemparameter, die erst nach Ersetzung dieses Makroaufrufes eingeführt wurden, können mit der Funktion SYSTEM-INFORMATION **nicht** abgefragt werden. Alle aktuell vorhandenen Systeminformationen und Systemparameter können mit den Kommandos SHOW-SYSTEM-INFORMATION und SHOW-SYSTEM-PARAMETERS abgefragt werden (unterstützen auch strukturierte Ausgabe in S-Variable).

Ergebnistyp: STRING

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

TASK-MODE( ) Task-Modus abfragen

Liefert den Modus der aktuellen Task.

Ergebnistyp: STRING (<string 2..6>)

TASK-MODE( )

TIME( ) Uhrzeit abfragen

Liefert die aktuelle Uhrzeit; das Trennzeichen zwischen den Angaben von Stunden, Minuten oder Sekunden ist frei wählbar.

Ergebnistyp: STRING (<string 8..8>)

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

TO-C-LITERAL( ) String in C-Literal konvertieren

Konvertiert den angegebenen String in ein C-Literal. Dabei wird jeweils an Anfang und Ende des Strings ein Hochkomma gesetzt. Hochkommas innerhalb des Strings werden verdoppelt. FROM-C-LITERAL( ) ist die inverse Funktion zu TO-C-LITERAL( ).

Ergebnistyp: STRING

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

TO-X-LITERAL( ) String in X-Literal konvertieren

Konvertiert den Hexadezimalwert des angegebenen Strings in die externe Darstellung: sie setzt an den Beginn und das Ende des Hexadezimalwerts jeweils ein Hochkomma und stellt außerdem das Zeichen X voran.

FROM-X-LITERAL( ) ist die inverse Funktion zu TO-X-LITERAL( ).

Ergebnistyp: STRING

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

TRANSLATE( ) Eingangswerten beliebige Ergebniswerte zuordnen

Mit der Funktion TRANSLATE( ) lassen sich beliebigen Eingangswerten beliebige Ergebniswerte zuordnen. Es dürfen bis zu zehn Umsetzungen angegeben werden, wobei WHEN- und THEN-Klauseln paarig auftreten müssen.

Ergebnistyp: STRING

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

TRANSLATE-BOOLEAN( ) Booleschen Ausdruck prüfen

Prüft, ob der Eingabe-Ausdruck wahr oder falsch ist. Ist er wahr, wird der in der THEN-Klausel angegebene Wert als Ergebnis geliefert. Ist der Eingabe-Ausdruck nicht wahr (=falsch), wird der in der ELSE-Klausel angegebene Wert als Ergebnis geliefert.

Ergebnistyp: BOOLEAN / INTEGER / STRING

TRANSLATE-BOOLEAN( )
IF = $ausdruck_1$,THEN = $ausdruck_2$,ELSE = $ausdruck_3$

TRIM( ) Gleiche Zeichen am Anfang oder Ende entfernen

Entfernt gleiche Zeichen am Anfang oder Ende oder an beiden Seiten eines Strings.

Ergebnistyp: STRING

TRIM( )
STRING = string_ausdruck ,SIDE = *BOTH / *LEFT / *RIGHT ,TRIM-BYTE = C'Ë' / zeichen

TSN( ) TSN abfragen

Liefert die Auftragsnummer (= Task Sequence Number) des aktuellen Auftrags.

Ergebnistyp: STRING (<string 4..4>)

TSN( )

UPPER-CASE( ) Kleinbuchstaben in Großbuchstaben umsetzen

Setzt alle Kleinbuchstaben im angegebenen String in Großbuchstaben um.

Die umzusetzenden Buchstaben müssen dem Standard-EBCDI-Code entsprechen. Es _werden keine sprachspezifischen Varianten unterstützt.

Ergebnistyp: STRING

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

USER-IDENTIFICATION( ) Benutzerkennung abfragen

Liefert die Benutzerkennung des aktuellen Auftrags, das heißt die Benutzerkennung aus dem SET-LOGON-PARAMETERS-Kommando.

Ergebnistyp: STRING (<string 1..8>)

USER-IDENTIFICATION( ) USER-ID( )

USER-SWITCH( ) Benutzerschalter auswerten

Prüft den Wert des angegebenen Benutzerschalters.

Benutzerschalter werden zum Beispiel zur Synchronisation von Stapelaufträgen eingesetzt, das heißt in Hintergrund-Prozeduren. Jeder Benutzerkennung stehen 32 Benutzerschalter zur Verfügung, die für alle unter der Benutzerkennung laufenden Aufträge gelten. Das heißt, Benutzerschalter, die in einem Auftrag gesetzt werden, können in einem anderen Auftrag, der unter der gleichen Benutzerkennung läuft, ausgewertet werden. Gesetzt werden Benutzerschalter mit dem Kommando MODIFY-USER-SWITCHES.

Ergebnistyp: BOOLEAN

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

VARIABLE-ATTRIBUTE( ) Variableneigenschaften abfragen

Liefert für die angegebene Variable den Wert des angegebenen Attributs.

Ergebnistyp: STRING

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

VARIABLE-TO-STRING( ) Variable konvertieren

Konvertiert eine S-Variable vom Typ Struktur in einen String (SDF-Kommando-String).

Ergebnistyp: STRING

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

VERIFY( ) Strings prüfen

Prüft zwei Strings und liefert als Ergebnis die Position des ersten Zeichens im String (STRING), das nicht im Suchstring (PATTERN) enthalten ist. Die Anzahl und Reihenfolge der Zeichen in den einzelnen Strings bleibt dabei unberücksichtigt. Die Suchrichtung ist frei wählbar, die Positionsangabe gilt immer ab Anfang des ersten Strings.

Ergebnistyp: INTEGER

VERIFY( )
STRING = string_ausdruck$_1$,PATTERN = string_ausdruck$_2$,DIRECTION = *FORWARD / *REVERSE

WILDCARD( ) Muster suchen

Vergleicht einen String mit einem Musterstring. Für den Musterstring gelten die allgemeinen Regeln für Muster im BS2000.

Ergebnistyp: BOOLEAN

WILDCARD( )
STRING = string_ausdruck1 ,PATTERN = string_ausdruck2 ,WILDCARD-MODE = *BS2000 / *POSIX

X-LITERAL-TO-INTEGER( ) String in Zahl konvertieren

Konvertiert einen maximal 4 Byte langen String in eine Dezimalzahl. Der Eingabestring kann als X-String oder C-String angegeben werden. Ein Leerstring erhält den Wert 0.
Besteht der Eingabestring aus weniger als 4 Zeichen, wird er von links nach rechts mit X'00' aufgefüllt.

X-LITERAL-TO-INTEGER( ) ist die inverse Funktion zu INTEGER-TO-X-LITERAL( ).

Ergebnistyp: INTEGER

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

SDF-P-Kommandos

Übersicht

Kommandos	Kurzbeschreibung
ASSIGN-STREAM	Weist einem S-Variablenstrom einen (Ausgabe-) Server zu
BEGIN-BLOCK	Leitet einen Kommandoblock ein
BEGIN-PARAMETER-DECLARATION	Leitet die Deklaration der Prozedurparameter im Prozedurkopf ein
BEGIN-STRUCTURE	Deklariert eine statische Struktur
CALL-PROCEDURE	Startet eine Kommandofolge
CLOSE-VARIABLE-CONTAINER	Schließt Variablenbehälter
COMPILE-PROCEDURE	Kompiliert eine Prozedur in ein Zwischenformat
CYCLE	Bricht einen Schleifendurchlauf ab
DECLARE-CONSTANT	Deklariert eine Variable und weist konstanten Wert zu
DECLARE-ELEMENT	Deklariert ein Strukturelement
DECLARE-PARAMETER	Deklariert Prozedurparameter
DECLARE-VARIABLE	Deklariert eine Variable
DELETE-STREAM	Löscht einen S-Variablenstrom
DELETE-VARIABLE	Löscht eine Variable
ELSE	Leitet den ELSE-Zweig im IF-Block ein
ELSE-IF	Leitet im IF-Block einen Alternativzweig ein
END-BLOCK	Schließt einen Kommandoblock ab
END-FOR	Schließt einen FOR-Block ab
END-IF	Schließt einen IF-Block ab
END-PARAMETER-DECLARATION	Schließt die Deklaration von Prozedurparametern ab
END-STRUCTURE	Schließt einen Strukturdeklarationsblock ab
END-WHILE	Schließt einen WHILE-Block ab
ENTER-PROCEDURE	Startet eine Prozedur im Hintergrund als Stapelauftrag
EXECUTE-CMD	Kommando ausführen und Ausgabe in Variable lenken
EXIT-BLOCK	Bricht die Verarbeitung eines Kommandoblocks ab
EXIT-PROCEDURE	Beendet eine Prozedur
FOR	Leitet einen FOR-Block ein
FREE-VARIABLE	Löscht den Inhalt einer Variablen

Kommandos	Kurzbeschreibung
GOTO	Springt zu einer Marke
IF	Leitet einen IF-Block ein
IF-BLOCK-ERROR	Leitet eine Block-Fehlerbehandlung ein
IF-CMD-ERROR	Leitet eine Kommando-Fehlerbehandlung ein
IMPORT-VARIABLE	Importiert eine Variable
INCLUDE-BLOCK	Führt BEGIN-BLOCK als Unterprozedur aus
INCLUDE-CMD	Ausführen einer Kommandofolge aus einem Programm
INCLUDE-PROCEDURE	Startet eine Kommandofolge als Include-Prozedur
MODIFY-PROCEDURE-OPTIONS	Ändert die Prozedureigenschaften während des Prozedurlaufs
MODIFY-PROCEDURE-TEST-OPTIONS	Ändert die Prozedurprotokollierung und begrenzt die Zahl der zulässigen Rückwärtssprünge
OPEN-VARIABLE-CONTAINER	Öffnet einen als PLAM-Bibliothekselement abgespeicherten Variablenbehälter
RAISE-ERROR	Erzeugt einen Returncode
READ-VARIABLE	Weist Variablen Werte zu
REPEAT	Leitet einen REPEAT-Block ein
REPEAT-CMD	Wiederholt eine Kommandoausführung
REPEAT-STMT	Wiederholt eine Anweisungsausführung
SAVE-RETURNCODE	Sichert den aktuellen Kommando-Returncode
SAVE-VARIABLE-CONTAINER	Sichert Variablenbehälter
SELECT-VARIABLE-ELEMENTS	Gibt die Elemente einer Variablen auf dem Bildschirm aus und schreibt davon ausgewählte zurück
SEND-DATA	Übergibt einen Datensatz an ein Programm
SEND-STMT	Übergibt einen Anweisungssatz an ein Programm
SET-PROCEDURE-OPTIONS	Legt Prozedureigenschaften fest
SET-VARIABLE	Weist einer Variablen einen Wert zu
SHOW-STREAM-ASSIGNMENT	Informiert über die aktuelle Zuweisung der S-Variablen-ströme
SHOW-STRUCTURE-LAYOUT	Informiert über die Elementnamen eines Strukturlayouts
SHOW-VARIABLE	Gibt Inhalte von Variablen aus
SHOW-VARIABLE-ATTRIBUTES	Informiert über die Attribute einer Variablen (Name, Wert, Datentyp, ...)

Kommandos	Kurzbeschreibung
SHOW-VARIABLE-CONTAINER-ATTR	Informiert über alle offenen Variablenbehälte
SORT-VARIABLE	Sortiert die Elemente einer Listenvariablen
TRACE-PROCEDURE	Setzt eine unterbrochene Prozedur schrittweise fort
TRANSMIT-BY-STREAM	Überträgt Variablen von der Client-Seite über einen S-Variablenstrom zu oder vom Server
UNTIL	Schließt einen REPEAT-Block ab
WHILE	Leitet einen WHILE-Block ein

Privilegien

Mit wenigen Ausnahmen können alle Kommandos von Benutzern mit den folgenden Privilegien aufgerufen werden:

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

Ausnahmen

– Kommando ENTER-PROCEDURE
erforderliche Privilegien: STD-PROCESSING oder HARDWARE-MAINTENANCE

– Benutzer mit den Privilegien SECURITY-ADMINISTRATION, SAT-FILE-MANAGEMENT oder SAT-FILE-EVALUATION können die folgenden Kommandos nur in Prozeduren 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

Kommando-Returncode

Alle SDF-P-Kommandos liefern Returncodes, die den Benutzer über die Ausführung des Kommandos informieren. Dieser Kommando-Returncode ist vergleichbar mit dem Returncode auf Programmebene. Der Kommando-Returncode ermöglicht es dem Benutzer, auf Fehlersituationen zu reagieren.

Der Kommando-Returncode besteht aus drei Teilen:

– dem Subcode1, der die aufgetretene Fehlersituation in eine Fehlerklasse einordnet, aus der abgeleitet werden kann, wie schwer wiegend ein Fehler ist. Der Wert von Subcode1 wird *dezimal* ausgegeben.
– dem Subcode2, der Zusatzinformationen zur Fehlerklasse enthalten kann.
– dem Maincode, der einem Meldungsschlüssel entspricht und differenzierte Fehlerinformationen liefert. Mit diesem Meldungsschlüssel kann über die vordefinierte Funktion MSG( ) oder mit dem Kommando HELP-MSG-INFORMATION die entsprechende Fehlermeldung ausgegeben werden.

Der Kommando-Returncode kann mit den vordefinierten Funktionen SUBCODE1( ), SUBCODE2( ) und MAINCODE( ) abgefragt werden.

Es gibt für jedes Kommando eigene Returncodes. Auch gibt es neben den eigenen Returncodes für bestimmte Kommandotypen noch übergreifende Returncodes, die hier anschließend aufgelistet werden.

Die allgemeinen Returncodes (d. h. Returncodes, die bei jedem Kommando auftreten können) sind:

(SC2)	SC1	Maincode	Bedeutung[1)]
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

[1)] Enthält die Tabelle auch garantierte Meldungen, wird die Bedeutungsspalte mit "/ garantierte Meldungen" ergänzt".

Bei allen Kommandos, Anweisungen und Datensätzen in denen eine Ausdrucksersetzung stattfindet, können die folgenden Returncodes auftreten, wenn bei der Ausdrucksersetzung Fehler passieren:

(SC2)	SC1	Maincode	Bedeutung
	1	SDP0140	Syntaxfehler während der Ersetzung
	64	SDP0141	Semantikfehler während der Ersetzung

Bei allen Datenzeilen in falschem Kontext kann folgender Returncode auftreten:

(SC2)	SC1	Maincode	Bedeutung
	64	SDP0091	Semantikfehler

Bei allen Anweisungen in falschem Kontext kann folgender Returncode auftreten:

(SC2)	SC1	Maincode	Bedeutung
	64	SDP0091	Semantikfehler

Garantierte Meldung

Der Zusatz „garantierte Meldung“ in der Returncode-Tabelle bedeutet:

Für ausgewählte Meldungen werden der Meldungsschlüssel und die Inserts (Numerierung und inhaltliche Zuordnung) garantiert.

Sofern in Verbindung mit einem Returncode garantierte Meldungen existieren, werden die Meldungsnummern nach dem Bedeutungstext aufgelistet.

Kommandos

ASSIGN-STREAM
S-Variablenstrom zuweisen

Weist einen S-Variablenstrom für strukturierte Ausgaben einem (Ausgabe-)Server zu, der die weitere Verarbeitung des Variablenstroms steuert.

```
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>
```

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
2	0	SDP0531	Warnung vom Server; Prozess wird fortgesetzt
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	64	SDP0532	Server-Fehler; Kommando abgewiesen
	64	SDP0534	Interner Server-Fehler; Kommando abgebrochen. Server-Verbindung abgebrochen nach unerwartetem Ereignis oder bei mangelhaften oder fehlenden System-Ressourcen.
	130	SDP0099	Kein Adressraum mehr verfügbar

BEGIN-BLOCK
Kommandoblock einleiten

Kommandoblöcke, die als logische Einheit behandelt werden sollen, werden mit dem Kommando BEGIN-BLOCK eingeleitet und mit dem Kommando END-BLOCK abgeschlossen. Diese Kommandoblöcke werden auch als BEGIN-Blöcke bezeichnet.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

BEGIN-PARAMETER-DECLARATION
Deklaration der Prozedurparameter einleiten

Die Prozedurparameter werden im Prozedurkopf mit dem Kommando DECLARE-PARAMETER deklariert. Wenn das Kommando DECLARE-PARAMETER mehrfach aufgerufen werden soll, werden diese Aufrufe in einen Kommandoblock eingeschlossen, der mit dem Kommando BEGIN-PARAMETER-DECLARATION eingeleitet und mit dem Kommando END-PARAMETER-DECLARATION abgeschlossen wird.

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

Kommando-Returncode

Die Returncodes können nur auftreten, wenn das Kommando außerhalb des Prozedurkopfes verwendet wird. Fehler im Prozedurkopf erkennt SDF-P bei der Voranalyse und beendet den Prozeduraufruf.

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

BEGIN-STRUCTURE
Statische Struktur deklarieren

Wenn ein Strukturlayout deklariert wird, kennzeichnet BEGIN-STRUCTURE den Beginn der Deklaration des Strukturlayouts. Das Strukturlayout muss vor den statischen Strukturen, die sich darauf beziehen sollen, deklariert werden. Das Kommando END-STRUCTURE beendet die Deklaration des Strukturlayouts.

Wenn eine statische Struktur mit der Angabe *BY-SYSCMD deklariert wird , muss direkt auf das Kommando DECLARE-VARIABLE, in dem die Struktur deklariert wird, das Kommando BEGIN-STRUCTURE folgen. Es leitet die Elementdeklarationen ein.

BEGIN-**STRUC**TURE

NAME = ***NONE** / <structured-name 1..20>(...)
<structured-name 1..20>(...)
SCOPE = ***CURR**ENT / ***PROC**EDURE / ***TASK**(...)
***TASK**(...)
STATE = ***ANY** / ***NEW**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

CALL-PROCEDURE
Kommandofolge starten

Das Kommando CALL-PROCEDURE startet eine gespeicherte Kommandofolge (Prozedur). Bei Abarbeitung werden darin enthaltene symbolische Parameter durch die im Kommandoaufruf angegebenen Werte (Operand PROCEDURE-PARAMETERS) ersetzt.

Hinweis

Das Kommando ist Bestandteil von BS2000/OSD-BC (Stand der Beschreibung: V7.0). Der Aufrufer benötigt (im Gegensatz zu den SDF-P-Kommandos) die Privilegien STD-PROCESSING bzw. HARDWARE-MAINTENANCE.

CALL-PROCEDURE Kurzname: **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 = ***YES** / ***NO**

Kommando-Returncode

Die nachfolgenden Kommando-Returncodes werden nur zurückgegeben, wenn die aufgerufene Prozedur selbst keinen Kommando-Returncode liefert (z.B. EXIT-PROCEDURE wegen Fehlers nicht ausgeführt).
Kommando-Returncodes, deren Maincode mit "SSM" beginnt, werden nur bei Aufruf einer Nicht-S-Prozedur zurückgegeben.
Kommando-Returncodes, deren Maincode mit "SDP" beginnt, werden nur bei Aufruf einer S-Prozedur zurückgegeben.

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
2	0	SSM2058	Protokoll-Typ-Fehler
2	0	SSM2065	EOF auf Prozedurdatei, /END-PROC simuliert
	1	SSM2036	Unvollständiger Operand
	1	SSM2054	Symbolischer Operandenfehler
	1	SSM2055	Symbolischer Operandenfehler in /BEGIN-PROC
	1	SDP0138	Fehler bei Voranalyse der Text-Prozedur oder Objekt-Prozedur fehlerhaft
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0093	Nicht-S-Prozedur kann nur Element vom Typ J sein
	64	SDP0144	Fehler bei Parameterübertragung
	64	SSM2052	DVS-Fehler (Open-Fehler)
	64	SSM2053	keine SAM/ISAM-Datei oder Datei beginnt nicht mit /BEGIN-PROC bzw. /PROC
	64	SSM2056	Parameter von /CALL-PROC und /BEGIN-PROC passen nicht zusammen
	64	SSM2061	Fehler bei Zugriff auf Bibilothekselement
	64	SSM2064	Prozedurdatei kann nicht von entferntem Rechner geholt werden
	130	SDP0099	Kein Adressraum mehr verfügbar
xx	xx	xxxxxxx	sonstige Returncodes der aufgerufenen Prozedur

CLOSE-VARIABLE-CONTAINER
Variablenbehälter schließen

Mit dem Kommando CLOSE-VARIABLE-CONTAINER werden die angegebenen Variablenbehälter geschlossen.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

COMPILE-PROCEDURE
Prozedur kompilieren

Das Kommando COMPILE-PROCEDURE konvertiert eine S-Prozedur in eine kompilierte Prozedur, d.h. in ein Zwischenformat, das dann auch in Umgebungen benutzt werden kann, in denen das Subsystem SDF-P nicht zur Verfügung steht.

In kompilierten Prozeduren kann der volle Umfang von SDF-P benutzt werden. Das gilt auch, wenn diese Prozeduren in einer Umgebung gestartet werden, die nur SDF-P-BASYS enthält.

COMPILE-PROCEDURE

FROM-FILE = <filename 1..54 without-gen> / ***LIB**RARY-**ELEM**ENT(...)
  ***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 / ***UP**PER-**LIM**IT / <composed-name 1..24>
    ,**TYPE** = **J** / <alphanum-name 1..8>
,**TO-FILE** = <filename1..54without-gen> **/ *LIB**RARY-**ELEM**ENT(...) / ***DUMMY**
  ***LIB**RARY-**ELEM**ENT(...)
    LIBRARY = ***SAME** / <filename 1..54 without-gen>
    ,**ELEM**ENT = ***SAME**(...) / <composed-name 1..64>(...)
      ***SAME**(...)
        VERSION = ***SAME** / ***UP**PER-**LIM**IT / ***INCREMENT** / *__HIGH__EST-**EXIST**ING /
          <composed-name 1..24>
      <composed-name 1..64>(...)
        VERSION = ***SAME** **/ ***UP**PER-**LIM**IT / ***INCREMENT** / *__HIGH__EST-**EXIST**ING /
          <composed-name 1..24>
    ,**TYPE** = **SYSJ** / <alphanum-name 1..8>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0138	Fehler bei Voranalyse der Prozedur garantierte Meldung: SDP0138
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

CYCLE
Schleifendurchlauf abbrechen

Das Kommando CYCLE kann in Schleifenblöcken (FOR-, WHILE-, REPEAT-Block) aufgerufen werden. Es bricht dann den aktuellen Schleifendurchlauf ab und setzt den Prozedurlauf mit der Ausführung des Abschlusskommandos des Schleifenblocks (END-FOR, END-WHILE, UNTIL) fort. Anschließend wird die Schleifenbedingung erneut überprüft und wenn nötig, der nächste Schleifendurchlauf gestartet.
Die Ausführung des Kommandos CYCLE kann von einer Bedingung abhängig gemacht werden.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)
	64	SDP0091	Semantikfehler (unkorrekter Ausdruck)
	130	SDP0099	Kein Adressraum mehr verfügbar

DECLARE-CONSTANT
Variable mit konstantem Wert deklarieren

Das Kommando DECLARE-CONSTANT deklariert eine oder mehrere Variablen und weist ihnen einen konstanten Wert zu. So sind diese Werte vor Überschreiben geschützt. Variablen mit konstantem Wert können nahezu so behandelt werden wie herkömmliche Variablen. Allerdings kann der Wert nicht mit SET-VARIABLE geändert und auch nicht mit FREE-VARIABLE entfernt werden.

DECLARE-**CONSTANT**

VARIABLE-**NAME** = list-poss(2000): <structured-name 1..20>(...)
<structured-name 1..20>(...)
VALUE = <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(...)
VARIABLE**-NAME** = <structured-name 1..20>
,**SCOPE** = ***VISIBLE** / ***TASK**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; Element schon deklariert
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler garantierte Meldungen: SDP1018, SDP1030
	130	SDP0099	Kein Adressraum mehr verfügbar

DECLARE-ELEMENT
Strukturelement deklarieren

Strukturelemente können einfache oder zusammengesetzte Variablen (Arrays, Strukturen, Listen) sein.
Variablenattribute, die im Kommando DECLARE-ELEMENT nicht definiert werden können, werden von der übergeordneten Struktur übernommen (z.B. SCOPE-Attribute von BEGIN-STRUCTURE oder DECLARE-VARIABLE).

Ist das Strukturelement eine zusammengesetzte Variable, müssen deren Elemente einzeln initialisiert werden. Zusammengesetzte Variablen können nicht als Ganzes initialisiert werden. Es können auch Elemente von dynamischen Strukturen mit diesem Kommando deklariert werden.

DECLARE-**ELEM**ENT

NAME = list-poss(2000): <composed-name 1..255>(...)
<composed-name 1..255>(...)
INITIAL-**VAL**UE = ***NONE** / <text 0..1800 with-low *expr*>
,**TYPE** = ***ANY** / ***STRING** / ***INTEGER** / ***BOOL**EAN / ***STRUC**TURE(...)
***STRUC**TURE(...)
DEFINITION = ***DYN**AMIC / ***BY-SYSCMD** / <structured-name 1..20>
,**MULT**IPLE-**ELEM**ENTS = ***NO** / ***ARRAY**(...) / ***LIST**(...)
***ARRAY**(...)
LOWER-**BOUND** = **0** / ***NONE** / <integer -2147483648..2147483647>
,**UP**PER-**BOUND** = ***NONE** / <integer -2147483648..2147483647>
***LIST**(...)
LIMIT = ***NONE** / <integer 1..2147483647>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; Element schon deklariert
2	0	CMD0001	Warnung; Operand INITIAL-VALUE wurde ignoriert
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler garantierte Meldung: SDP1018
	130	SDP0099	Kein Adressraum mehr verfügbar

DECLARE-PARAMETER
Prozedurparameter deklarieren

Mit dem Kommando DECLARE-PARAMETER werden die Prozedurparameter deklariert, die während des Prozedurablaufs eine (konkreten) Wert benötigen. Des Weiteren wird die Art und Weise der Übergabe der Parameterwerte an die Prozedur vereinbart (Anfangswert, Prompting, ...). Die Deklaration von Prozedurparametern ist nur im Prozedurkopf erlaubt.

Prozedurparameter sind in SDF-P prozedurlokale Variablen: Bei der Definition im Prozedurkopf erhalten sie implizit den Geltungsbereich SCOPE = *CURRENT.
Die Namen der Prozedurparameter sind gleichzeitig die Schlüsselwörter der Prozedurparameter im Operanden PROCEDURE-PARAMETERS der Kommandos CALL-, ENTER- und INCLUDE-PROCEDURE.

DECLARE-**PAR**AMETER

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

<structured-name 1..20>(...)

INITIAL-**VAL**UE = ***NONE** / ***PROMPT(...)** / <text 0..1800 with-low *expr*>

***PROMPT**(...)

PROMPT-STRING = ***STD** / <text 0..1800 with-low *string-expr*>

,DEFAULT-**VAL**UE = ***NONE** / <text 0..1800 with-low *expr*>

,SECRET-INPUT = ***NO** / ***Y**ES

,**TYPE** = ***ANY** / ***STRING** / ***INTEGER** / ***BOOL**EAN

,**TRANS**FER-**TYPE** = ***BY-VAL**UE / ***BY-REF**ERENCE

Kommando-Returncode

Nur wenn DECLARE-PARAMETER in einem anderen (d.h. falschen) Kontext benutzt wird, erscheinen die folgenden Returncodes:

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

DECLARE-VARIABLE
Variable deklarieren

Mit DECLARE-VARIABLE werden die Attribute dieser Variablen und evtl. auch ein Anfangswert festgelegt.
Die Einbindung von Jobvariablen in SDF-P ist über den Operanden CONTAINER möglich.

DECLARE-**VAR**IABLE Kurzname: **DCV**

VARIABLE**-NAME** = list-poss(2000): <structured-name 1..20>(...)
<structured-name 1..20>(...)
INITIAL**-VAL**UE = ***NONE** / <text 0..1800 with-low *expr*>
,**TYPE** = ***ANY** / ***STRING** / ***INTEGER** / ***BOOL**EAN / ***STRUC**TURE(...)
***STRUC**TURE(...)
DEFINITION = ***DYN**AMIC / ***BY-SYSCMD** / <structured-name 1..20>
,**MULT**IPLE**-ELEM**ENTS = ***NO** / ***ARRAY**(...) / ***LIST**(...)
***ARRAY**(...)
LOWER**-BOUND** = **0** / ***NONE** / <integer -2147483648..2147483647>
,**UP**PER**-BOUND** = ***NONE** / <integer -2147483648..2147483647>
***LIST**(...)
LIMIT = ***NONE** / <integer 1..2147483647>
,**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(...) / ***JV**(...)
***VAR**IABLE(...)
VARIABLE**-NAME** = <structured-name 1..20>
,**SCOPE** = ***VISIBLE** / ***TASK**
***JV**(...)
JV-NAME = <filename 1..54>
,**STATE** = ***ANY** / ***NEW** / ***OLD**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; Element schon deklariert
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler garantierte Meldungen: SDP1018, SDP1030
	130	SDP0099	Kein Adressraum mehr verfügbar

DELETE-STREAM
S-Variablenstrom löschen

Das Kommando DELETE-STREAM löscht S-Variablenströme. Ihre Zuweisungen werden nicht mehr durch das Kommando SHOW-STREAM-ASSIGNMENT angezeigt.
Es werden nur Ströme gelöscht, die auf der obersten Prozedurebene (im Dialog-Modus) erstellt wurden. Weitere Übertragungen über den gelöschten Strom werden zurückgewiesen.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
2	0	SDP0516	Angegebener Variablenstrom-Name existiert nicht oder das Suchmuster wurde nicht gefunden; Prozess wird fortgesetzt
1	0	SDP0518	Kein Treffer bei Wildcard. Prozess wird fortgesetzt
2	0	SDP0535	Warnung vom Server bei der Löschung eines bestimmten S-Variablenstroms; Prozess wird fortgesetzt
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	64	SDP0515	Angegebener Variablenstrom nicht auf oberster Ebene erzeugt
	64	SDP0536	Server-Fehler; das Löschen des bestimmten S-Variablenstroms wurde abgewiesen.
	64	SDP0537	Interner Server-Fehler; das Löschen des bestimmten S-Variablenstroms wurde abgewiesen. Server-Verbindung abgebrochen nach unerwartetem Ereignis oder bei mangelhaften oder fehlenden System-Ressourcen

DELETE-VARIABLE
Variable löschen

DELETE-VARIABLE löscht die Deklaration einer S-Variablen innerhalb des aktuellen Geltungsbereichs, d.h. auch Deklarationen von importierten Task-Variablen.
Es können einfache und zusammengesetzte Variablen gelöscht werden, aber nicht einzelne Elemente von zusammengesetzten Variablen.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; keine Aktion durchgeführt
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

ELSE
ELSE-Zweig im IF-Block einleiten

Im IF-Block leitet das ELSE-Kommando den letzten Zweig ein. Die Kommandos zwischen ELSE-Kommando und dem Abschlusskommando END-IF werden ausgeführt, wenn keine der zuvor in IF- oder ELSE-IF-Kommandos geprüften Bedingungen zutrifft.
Im IF-BLOCK-ERROR- und IF-CMD-ERROR-Block wird der ELSE-Zweig durchlaufen, wenn kein Fehler aufgetreten ist.

ELSE

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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
Im IF-Block einen Alternativzweig einleiten

Der ELSE-IF-Zweig wird ausgeführt, wenn die im ELSE-IF-Kommando angegebene Bedingung zutrifft; andernfalls wird der nächste Zweig des IF-Blocks bzw. ein END-IF-Kommando gesucht. Zum ELSE-IF-Zweig gehören alle Kommandos, die zwischen dem aktuellen ELSE-IF-Kommando und den nachfolgenden ELSE-IF-, ELSE- oder END-IF-Kommando stehen.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

END-BLOCK
Kommandoblock abschließen

END-BLOCK schließt einen BEGIN-Block ab, das heißt einen Kommandoblock, der mit dem Kommando BEGIN-BLOCK eingeleitet wurde.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

END-FOR
FOR-Block abschließen

END-FOR schließt einen FOR-Block ab, d. h. eine FOR-Schleife, die mit dem FOR-Kommando eingeleitet wurde.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0139	Back Branch-Grenze erreicht
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

END-IF
IF-Block abschließen

END-IF schließt Blöcke mit bedingten Kommandofolgen ab. Das sind:
– IF-Block
– IF-BLOCK-ERROR-Block
– IF-CMD-ERROR-Block

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

END-PARAMETER-DECLARATION
Prozedurparameterdeklaration abschließen

END-PARAMETER-DECLARATION beendet den Kommandoblock, der mit dem Kommando BEGIN-PARAMETER-DECLARATION eingeleitet wurde; in diesem Block werden die Prozedurparameter deklariert

END-PARAMETER-**DECL**ARATION

Kommando-Returncode

Nur wenn END-PARAMETER-DECLARATION in einem anderen (d.h. falschen) Kontext benützt wird, erscheinen die folgenden Returncodes:

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

END-STRUCTURE
Ende einer Strukturdeklaration kennzeichnen

END-STRUCTURE schließt einen Strukturdeklarationsblock ab, der mit dem Kommando BEGIN-STRUCTURE eingeleitet wurde.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
2	0	CMD0001	Warnung; Struktur ist leer
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

END-WHILE
WHILE-Block abschließen

END-WHILE schließt einen WHILE-Block ab, d.h. eine Schleife, die mit dem WHILE-Kommando eingeleitet wurde.
Bei der Ausführung des END-WHILE-Kommandos wird die Schleifenbedingung im WHILE-Kommando überprüft. Falls sie erfüllt ist (TRUE), wird mit dem ersten Kommando des WHILE-Blocks der nächste Schleifendurchgang gestartet. Andernfalls wird die Schleife beendet und der Prozedurlauf mit dem ersten Kommando fortgesetzt, das auf END-WHILE folgt.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0139	Back Branch-Grenze erreicht
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

ENTER-PROCEDURE
Prozedur im Hintergrund als Stapelauftrag starten

Mit dem Kommando ENTER-PROCEDURE kann der Benutzer eine Prozedur als Stapelauftrag starten.
Das Kommando ist Bestandteil von BS2000/OSD (Stand der Beschreibung: V7.0).

(Teil 1 von 2)

ENTER-PROCEDURE Kurzname: **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 = **<u>*NO</u>** / <text 0..1800 with-low>
,**PROC**ESSING-**ADMIS**SION = **<u>*SAME</u>** / ***PAR**AMETERS(...)
  ***PAR**AMETERS(...)
    USER-IDENTIFICATION = **<u>*NONE</u>** / <name 1..8>
    ,**ACCOUNT** = **<u>*NONE</u>** / <alphanum-name 1..8>
    ,**PASS**WORD = **<u>*NONE</u>** / <c-string 1..8> / <c-string 9..32> / <x-string 1..16> / ***SECRET**
,**PROC**EDURE-**PASS**WORD = **<u>*NONE</u>** / <x-string 1..8> / <c-string 1..4> /
  <integer -2147483648..2147483647> / ***SECRET**
,**CRYPTO-PASS**WORD = **<u>*NONE</u>** / <c-string 1..8> / <x-string 1..16> / ***SECRET**
,**HOST** = **<u>*STD</u>** / <c-string 1..8> / ***ANY**
,**JOB-CLASS** = **<u>*STD</u>** / <name 1..8>
,**JOB-NAME** = **<u>*NO</u>** / <name 1..8>
,**MONJV** = **<u>*NONE</u>** / <filename 1..54 without-gen-vers>
,**JV-PASS**WORD = **<u>*NONE</u>** / <c-string 1..4> / <x-string 1..8> / ***SECRET** /
  <integer -2147483648..2147483647>
,**JOB-PRIO**RITY = **<u>*STD</u>** / <integer 1..9>
,**RERUN-AF**TER-**CRASH** = **<u>*NO</u>** / ***Y**ES
,**FLUSH-AF**TER-**SHUTDOWN** = **<u>*NO</u>** / ***Y**ES
,**SCHED**ULING-**TIME** = **<u>*STD</u>** / ***PAR**AMETERS(...) / ***BY-CALENDAR**(...)
  ***PAR**AMETERS(...)
    START = **<u>*STD</u>** / ***SOON** / ***IMMED**IATELY / ***AT-STREAM**-STARTUP / ***WITHIN**(...) / ***AT**(...) /
      ***EARL**IEST(...) / ***LATEST**(...)
      ***WITHIN**(...)
        HOURS = **<u>0</u>** / <integer 0..23 *hours*>
        ,**MINUT**ES = **<u>0</u>** / <integer 0..59 *minutes*>
      ***AT**(...)
        DATE = **<u>*TODAY</u>** / <date>
        ,**TIME** = <time>

Fortsetzung ➠

(Teil 2 von 2)

```
        *EARLIEST(...)
            DATE = *TODAY / <date>
            ,TIME = <time>
        *LATEST(...)
            DATE = *TODAY / <date>
            ,TIME = <time>
    ,REPEAT-JOB = *STD / *NO / *DAILY / *WEEKLY / *AT-STREAM-STARTUP / *PERIOD(...)
        *PERIOD(...)
            HOURS = 0 / <integer 0..23 hours>
            ,MINUTES = 0 / <integer 0..59 minutes>
*BY-CALENDAR(...)
    CALENDAR-NAME = <filename 1..54 without-gen-vers>
    ,SYMBOLIC-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 = *PARAMETERS (...)
*PARAMETERS(...)
    RUN-PRIORITY = *STD / <integer 30..255>
    ,CPU-LIMIT = *STD / *NO / <integer 1..32767 seconds>
    ,SYSLST-LIMIT = *STD / *NO / <integer 0..999999>
    ,SYSOPT-LIMIT = *STD / *NO / <integer 0..999999>
,LOGGING = *STD / *YES / *NO
,LISTING = *NO / *YES
,JOB-PARAMETER = *NO / <c-string 1..127>
,SYSTEM-OUTPUT = *STD / *PRINT / *DELETE
,ASSIGN-SYSTEM-FILES = *STD / *PARAMETERS(...)
*PARAMETERS(...)
    SYSLST = *STD / *PRIMARY / *DUMMY / <filename 1..54>
    ,SYSOUT = *STD / *PRIMARY / *DUMMY / <filename 1..54>
,PROTECTION = *NONE / *CANCEL
```

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Kommando ausgeführt
2	0	CMD0002	Kommando ausgeführt mit Warnung z.B. DELETE= *YES ignoriert für Repeatjob oder bei Angabe eines Bibliothekelements
	1	CMD0202	Syntaxfehler im Kommando
	32	CMD0221	Systemfehler
	64	JMS0630	Semantik- oder Privilegienfehler z.B. Rechner, Katalogkennung, Jobklasse unbekannt; MONJV nicht zugreifbar
	64	JMS0640	Fehler bei Zugriff auf die Prozedurdatei z.B. keine SAM- bzw. ISAM-Datei, Datei leer, fehlendes Zugriffsrecht
	64	JMS0670	Fehler bei einem REMOTE-Auftrag
	130	JMS0620	Kein weiterer Speicherplatz oder TSN verfügbar; oder angegebene MONJV überwacht bereits einen Auftrag
	130	JMS0650	MSCF nicht verfügbar oder keine Verbindung zum angegebenen Rechner

EXECUTE-CMD
Kommando ausführen und strukturierte Ausgabe

Das Kommando EXECUTE-CMD übergibt das beim Operanden CMD=... angegebene Kommando zur Ausführung und schreibt die Kommandoausgaben (Meldungen, Ausgabeinformation) in eine anzugebende Variable. Es können unstrukturierte und strukturierte Ausgaben erzeugt werden, je nach Leistung des Kommandoservers. Das Verhalten bei fehlerhafter Kommandoausführung kann durch Auswerten des Returncodes gesteuert werden.

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>

Kommando-Returncode

*Bei RETURNCODE = *STD ist möglich:*

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar
xx	xx	xxxxxxx	sonstige Returncodes des ausgeführten Kommandos

*Bei Returncode = *NONE / *VARIABLE(...) ist möglich:*

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler (aber nur im EXECUTE-CMD)
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

EXIT-BLOCK
Verarbeitung eines Kommandoblocks abbrechen

EXIT-BLOCK bricht die Verarbeitung eines Kommandoblocks ab (BEGIN-, FOR-, IF-, WHILE-, REPEAT-Block etc.) und setzt den Prozedurlauf mit dem Kommando fort, das auf das Blockabschlusskommando folgt. Bei einem BEGIN-Block, der mit einem INCLUDE-BLOCK-Kommando aufgerufen wurde, wird der Prozedurlauf mit dem Kommando fortgesetzt, das auf das INCLUDE-BLOCK-Kommando folgt.
Die Ausführung des Kommandos EXIT-BLOCK kann von einer Bedingung abhängig gemacht werden.

EXIT-BLOCK

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

,**COND**ITION = ***NONE** / <text 1..1800 with-low *bool-expr*>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)
	64	SDP0091	Semantikfehler (unkorrekter Ausdruck)
	130	SDP0099	Kein Adressraum mehr verfügbar

EXIT-PROCEDURE
Prozedur beenden

EXIT-PROCEDURE beendet die aktuelle Prozedur und gibt die Ablaufsteuerung an die aufrufende Ebene zurück.

Die Angabe ERROR=*YES stößt die Fehlerbehandlung an, wenn SUBCODE1 nicht Null ist.

```
EXIT-PROCEDURE

ERROR = *NO (...) / *YES(...)
    *NO(...)
        SUBCODE2 = 0 / < integer 0..255>
        ,MAINCODE = CMD0001 / <alphanum-name 7..7>
    *YES(...)
        SUBCODE1 = 64 / <integer 0..255>
        ,SUBCODE2 = 0 / <integer 0..255>
        ,MAINCODE = SDP0018 / <alphanum-name 7..7>
,RESUME-PROGRAM = *NO / *YES
```

Kommando-Returncode

Wenn die Ausführung des EXIT-PROCEDURE Kommandos selbst zu einem Fehler führt, wird nicht zum Aufrufer zurückgekehrt, sondern einer der folgenden Returncodes geliefert:

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

FOR
FOR-Block einleiten

Das Kommando FOR leitet einen FOR-Block ein; das Kommando /END-FOR beendet einen FOR-Block. Das Konstrukt ist eine FOR-Schleife.

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*>
| ,**INCR**EMENT = **1** / <text 0..1800 *arithm-expr*>

<text 0..1800 with-low *expr*>

,**COND**ITION = ***NONE** / <text 0..1800 with-low *bool-expr*>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

FREE-VARIABLE
Inhalt einer Variablen löschen

Das Kommando FREE-VARIABLE löscht den Inhalt einer oder mehrerer S-Variablen.

Das Kommando FREE-VARIABLE kann auf einfache und zusammengesetzte Variable angewendet werden. Bei einer Listenvariablen können gezielt ein oder mehrere Elemente gelöscht werden. Bei einer Variablen mit dynamischer Struktur werden nicht nur die Inhalte der Elemente gelöscht, sondern auch die Elemente selbst.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; keine Elemente gelöscht; Variable ist bereits gelöscht
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler garantierte Meldung: SDP1008
	130	SDP0099	Kein Adressraum mehr verfügbar

GOTO
Zu einer Marke springen

GOTO springt zu der angegebenen Marke und setzt dort den Prozedurlauf fort. Die Marke muss im gleichen oder in einem umgebenden Block definiert sein (siehe Beispiel 2). Außerdem muss sie innerhalb der Prozedur eindeutig sein.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0139	Back Branch-Grenze erreicht
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

IF
IF-Block einleiten

Das Kommando IF leitet einen IF-Block, das heißt eine bedingte Kommandofolge ein. Der IF-Block muss mit dem Kommando END-IF abgeschlossen werden.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

IF-BLOCK-ERROR
Block-Fehlerbehandlung einleiten

IF-BLOCK-ERROR leitet eine Block-Fehlerbehandlung ein.
Im IF-BLOCK-ERROR-Block kann mit dem Kommando ELSE ein ELSE-Zweig definiert werden. Abgeschlossen wird der IF-BLOCK-ERROR-Block mit dem Kommando END-IF.

IF-BLOCK-ERROR

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

IF-CMD-ERROR
Kommando-Fehlerbehandlung einleiten

IF-CMD-ERROR leitet eine Kommandofolge ein, die durchlaufen wird, wenn im unmittelbar vorhergehenden Kommando ein Fehler auftritt. Hiermit ist eine gezielte Fehlerbehandlung für dieses Kommando möglich und eine Blockfehlerbehandlung wird somit verhindert. Der IF-CMD-ERROR-Block muss mit dem Kommando END-IF abgeschlossen werden.

IF-CMD-ERROR

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

IMPORT-VARIABLE
Variable importieren

Mit dem Kommando IMPORT-VARIABLE kann eine zuvor deklarierte Variablen in die gerufene Prozedur importiert werden. IMPORT-VARIABLE entspricht so in gewisser Weise dem Kommando DECLARE-VARIABLE, erspart aber die wiederholte Aufzählung aller zugewiesenen Attribute.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; Element schon deklariert
2	0	SDP2000	Warnung: Nicht alle Elemente der Eingabeliste konnten erfolgreichabgearbeitet werden. garantierte Meldung: SDP2000
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler; garantierte Meldungen: SDP1008, SDP1018
	64	SDP2001	Keines der Elemente konnte importiert werden
	130	SDP0099	Kein Adressraum mehr verfügbar

INCLUDE-BLOCK
BEGIN-Block als Unterprozedur ausführen

Das Kommando INCLUDE-BLOCK verzweigt zu dem Kommando BEGIN-BLOCK mit der angegebenen Marke (Namen). Nach Ausführung des (nächsten) Kommandos END-BLOCK oder EXIT-BLOCK erfolgt der Rücksprung zu dem Kommando, das dem Kommando INCLUDE-BLOCK folgt.
Das Kommando INCLUDE-BLOCK ermöglicht die komfortable Ausführung einer Unterprozedur, die zwischen den Kommandos BEGIN-BLOCK und END-BLOCK definiert ist.

Das Kommando INCLUDE-BLOCK wird im Dialog-Modus abgewiesen.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)

INCLUDE-CMD
Kommandofolgen aus Programmen aufrufen

Mit dem Kommando INCLUDE-CMD können Kommandos und Kommandofolgen aus einem Programm heraus zur Ausführung übergeben werden. Das Kommando kann nur im Makro CMD (TU-Programm) und in EXECUTE-SYSTEM-CMD-Anweisungen ausgeführt werden.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0138	Fehler während der Prozedur-Voranalyse
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

INCLUDE-PROCEDURE
Kommandofolge aus Include-Prozedur starten

Das Kommando INCLUDE-PROCEDURE startet eine gespeicherte Kommandofolge (Prozedur). Eine Include-Prozedur ist eine Prozedur, in der die Datenumgebung der aufrufenden Prozedur sichtbar ist. Bei Abarbeitung werden darin enthaltene symbolische Parameter durch die im Kommandoaufruf angegebenen Werte (Operand PROCEDURE-PARAMETERS) ersetzt.

INCLUDE-**PROC**EDURE Kurzname: **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
```

Kommando-Returncode

Die nachfolgenden Kommando-Returncodes können nur zurückgegeben werden, wenn die aufgerufene Prozedur selbst keinen Kommando-Returncode liefert (z.B. EXIT-PROCEDURE wegen Fehlers nicht ausgeführt).
Kommando-Returncodes, deren Maincode mit "SSM" beginnt, können nur bei Aufruf einer Nicht-S-Prozedur zurückgegeben werden.
Kommando-Returncodes, deren Maincode mit "SDP" beginnt, können nur bei Aufruf einer S-Prozedur zurückgegeben werden.

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
2	0	SSM2058	Protokoll-Typ-Fehler
2	0	SSM2065	EOF auf Prozedurdatei, /END-PROC simuliert
	1	SSM2036	Unvollständiger Operand
	1	SSM2054	Symbolischer Operandenfehler

Fortsetzung ➠

(SC2)	SC1	Maincode	Bedeutung
	1	SSM2055	Symbolischer Operandenfehler in /BEGIN-PROC
	1	SDP0138	Fehler bei Voranalyse der Text-Prozedur oder Objekt-Prozedur fehlerhaft
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0093	Nicht-S-Prozedur kann nur Element vom Typ J sein
	64	SDP0144	Fehler bei Parameterübertragung
	64	SSM2052	DVS-Fehler (Open-Fehler)
	64	SSM2053	keine SAM/ISAM-Datei oder Datei beginnt nicht mit /BEGIN-PROC bzw. /PROC
	64	SSM2056	Parameter von /INCL-PROC und /BEGIN-PROC passen nicht zusammen
	64	SSM2061	Fehler bei Zugriff auf Bibilothekselement
	64	SSM2064	Prozedurdatei kann nicht von entferntem Rechner geholt werden
	130	SDP0099	Kein Adressraum mehr verfügbar
xx	xx	xxxxxxx	sonstige Returncodes der aufgerufenen Prozedur

MODIFY-PROCEDURE-OPTIONS
Prozedureigenschaften während des Prozedurlaufs ändern

Mit MODIFY-PROCEDURE-OPTIONS können die meisten Prozedureigenschaften, die zu Prozedurbeginn mit dem Kommando SET-PROCEDURE-OPTIONS festgelegt wurden, während des Prozedurlaufs geändert werden.

MODIFY-**PROC**EDURE-**OPT**IONS

IMPLICIT-**DECL**ARATION = ***UNCHA**NGED / ***Y**ES / ***NO**
,**LOG**GING-**ALLOW**ED = ***PAR**AMETERS(...) / ***NO** / ***Y**ES
  ***PAR**AMETERS(...)
    CMD = ***UNCHA**NGED / ***Y**ES / ***NO**
    ,**DATA** = ***UNCHA**NGED / ***Y**ES / ***NO**
,**INTERRUPT-ALLOW**ED = ***UNCHA**NGED / ***Y**ES / ***NO**
,**DATA-ESC**APE-**CHAR** = ***UNCHA**NGED / ***NONE** / '&&' / '#' / '*' / '@' / '$' / ***STD**
,**DATA-ERROR**-HANDLING = ***UNCHA**NGED / ***Y**ES / ***NO**
,**JV-REPL**ACEMENT = ***UNCHA**NGED / ***NONE** / ***AFTER-BUILTIN-FUNCTION**
,**ERROR-MECH**ANISM = ***UNCHA**NGED / ***SPIN-OFF-COMPATIBLE** / ***BY-RETURNCODE**
,**SUP**PRESS-**SDP-MSG** = ***UNCHA**NGED / ***NONE** / ***ADD(**...) / ***REM**OVE**(**...)
  ***ADD**(...)
    MSG-ID = list-poss(2000): <alphanum-name 7..7>)
  ***REM**OVE(...)
    MSG-ID = list-poss(2000): <alphanum-name 7..7>)

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

MODIFY-PROCEDURE-TEST-OPTIONS
Protokollierung ändern, Rückwärtssprünge begrenzen

Mit dem Kommando MODIFY-PROCEDURE-TEST-OPTIONS können für Testzwecke folgende Einstellungen zum Prozedurablauf geändert werden:

– Protokollierung von Kommandos und Daten
– Begrenzen von Rückwärtssprüngen
– Simulieren von SDF-P-BASYS-Verhalten

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**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	64	SDP0091	Semantikfehler
	64	SDP0133	Unvollständiger Datentyp beim Ausdruck
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

OPEN-VARIABLE-CONTAINER
Variablenbehälter öffnen

Mit dem Kommando OPEN-VARIABLE-CONTAINER werden Variablenbehälter, die als PLAM-Bibliothekselemente abgespeichert sind, geöffnet. Existiert ein solcher Variablenbehälter bzw. ein solches Element beim Aufruf dieses Kommandos noch nicht, wird er bzw. es automatisch erzeugt.

```
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>
```

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
2	0	SDP00xx	Warnung, weil folgendes passiert ist: garantierte Meldungen: SDP1008, SDP1018
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

RAISE-ERROR
Returncode erzeugen

RAISE-ERROR erzeugt einen Kommando-Returncode und stößt anschließend die Fehlerbehandlung an, wenn SUBCODE 1 nicht Null ist.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar
xx	xx	xxxxxxx	Returncode wie in Operanden übergeben

READ-VARIABLE
Variablen Werte zuweisen

Das Kommando READ-VARIABLE liest Daten satzweise von einem Eingabemedium ein und speichert sie in einer Variablen ab. Der Vorgang wird beendet, wenn der String "*END-OF-CMD" oder Dateiende (EOF) auftritt.

READ-VARIABLE Kurzname: **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** = ***OPT**IONAL / ***Y**ES / ***NO**

,**INPUT** = ***TER**MINAL(...) / <filename 1..54 without-gen-vers>(...) / ***VAR**IABLE(...) / ***LIB**RARY-**ELEM**ENT(...) / ***SYSDTA**(...)

***TER**MINAL(...)

PROMPT-STRING = '>>' / <text 0..1800 with-low>

,**SECRET-INPUT** = ***NO** / ***Y**ES

<filename 1..54 without-gen-vers>(...)

REMOVE-**KEY** = ***Y**ES / ***NO**

,**BEG**IN-**REC**ORD = ***FIRST** / <integer 1..2147483647>

,**END-REC**ORD = ***LAST** / <integer 1..2147483647>

,**BEG**IN-**COL**UMN = ***FIRST** / <integer 1..2147483647>

,**END-COL**UMN = ***LAST** / <integer 1..2147483647>

,**PATTERN** = ***NONE** / <c-string 0..1800 with-low>

,**PATTERN-TYPE** = ***STRING** / ***REGULAR-EXPRESSION**

***VAR**IABLE(...)

VARIABLE-**NAME** = <composed-name 1..255>

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

LIBRARY = <filename 1..54 without-vers>

,**ELEM**ENT = <composed-name 1..64>(...)

<composed-name 1..64>(...)

VERSION = ***HIGH**EST-**EXIST**ING / <composed-name 1..24>

,**TYPE** = **S** / <alphanum-name 1..8>

***SYSDTA**(...)

REMOVE-**KEY** = ***Y**ES / ***NO**

Kommando-Returncode

Es ist möglich, dass ein Teil des Kommandos schon abgearbeitet und ausgeführt wurde, bevor der Fehler auftrat. In diesem Fall ist das Ergebnis des Kommandos nicht garantiert.

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
2	0	SDP2000	Warnung: Nicht alle Elemente der Eingabeliste konnten erfolgreichabgearbeitet werden. garantierte Meldung: SDP2000
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0089	INPUT-Fehler
	64	SDP0091	Semantikfehler garantierte Meldung: SDP1008
	64	SDP2001	Keines der Elemente konnte eingelesen werden
	130	SDP0099	Kein Adressraum mehr verfügbar

REPEAT
REPEAT-Block einleiten

REPEAT leitet einen REPEAT-Block (REPEAT-Schleife) ein. Die Ausführung der Kommandofolge innerhalb des REPEAT-Blocks wird wiederholt, solange die Schleifenbedingung erfüllt ist. Die Schleifenbedingung wird im UNTIL-Kommando geprüft, das den REPEAT-Block abschließt.

REPEAT

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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

REPEAT-CMD
Kommandoausführung wiederholen

Das Kommando REPEAT-CMD ermöglicht die wiederholte Ausführung eines Kommandos, wobei ein speziell gekennzeichneter Teil (Namensteil, Operand, Operandenwert) nacheinander durch die Elemente einer Eingabeliste substituiert wird. Diese Elemente können als Sätze in einer Datei, einem Bibliothekselement oder als Listenelemente einer S-Variablen spezifiziert oder direkt am Terminal eingegeben werden. Das angegebene Kommando wird für jedes Element der Eingabeliste aufgerufen.

REPEAT-CMD Kurzname: **REPCMD**

CMD = <text 0..1800 with-low>
,**SUBST**ITUTION**-LIST** = ***TERMINAL** / <filename 1..54 without-gen-vers> / ***VAR**IABLE(...) /
***LIB**RARY**-ELEM**ENT(...)

***VAR**IABLE(...)
VARIABLE**-NAME** = <composed-name 1..255>

***LIB**RARY**-ELEM**ENT(...)
LIBRARY = <filename 1..54 without-vers>
,**ELEM**ENT = <composed-name 1..64>(...)
<composed-name 1..64>(...)
VERSION = **HIGH**EST-**EXIST**ING / <composed-name 1..24>
,**TYPE** = **S** / <alphanum-name 1..8>

,**DIALOG-SEL**ECTION = ***NO** / ***Y**ES
,**CONT**INUE**-AFTER-ERROR** = ***Y**ES / ***NO**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
2	0	SDP2000	Warnung: Nicht alle Elemente der Eingabeliste konnten erfolgreich abgearbeitet werden. garantierte Meldung: SDP2000
	1	SDP2001	Keines der Elemente der Eingabeliste konnte erfolgreich abgearbeitet werden. garantierte Meldung: SDP2001

REPEAT-STMT
Anweisungsausführung wiederholen

Das Kommando REPEAT-STMT ermöglicht die wiederholte Ausführung einer Anweisung (SDF-Format), wobei ein speziell gekennzeichneter Teil (Namensteil, Operand, Operandenwert) nacheinander durch die Elemente einer Eingabeliste substituiert wird. Diese Elemente können als Sätze in einer Datei, einem Bibliothekselement oder als Listenelemente einer S-Variablen spezifiziert oder direkt am Terminal eingegeben werden. Die angegebene Anweisung wird für jedes Element der Eingabeliste aufgerufen.

REPEAT-STMT Kurzname: **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(...)
VARIABLE**-NAME** = <composed-name 1..255>
***LIB**RARY**-ELEM**ENT(...)
LIBRARY = <filename 1..54 without-vers>
,**ELEM**ENT = <composed-name 1..64>(...)
<composed-name 1..64>(...)
VERSION = ***HIGH**EST-**EXIST**ING / <composed-name 1..24>
,**TYPE** = **S** / <alphanum-name 1..8>
,**DIALOG-SEL**ECTION = ***NO** / ***Y**ES
,**CONT**INUE**-AFTER-ERROR** = *YES / ***NO**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
2	0	SDP2000	Warnung: Nicht alle Elemente der Eingabeliste konnten erfolgreich abgearbeitet werden. garantierte Meldung: SDP2000
	1	SDP2001	Keines der Elemente der Eingabeliste konnte erfolgreich abgearbeitet werden. garantierte Meldung: SDP2001

SAVE-RETURNCODE
Aktuellen Kommando-Returncode sichern

SAVE-RETURNCODE sichert den aktuellen Kommando-Returncode, so dass er mit SDF-P-Funktionen ausgewertet werden kann. Zur Auswertung des Returncodes stehen die vordefinierten Funktionen SUBCODE1( ), SUBCODE2( ) und MAINCODE( ) zur Verfügung.

SAVE-RETURNCODE

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	130	SDP0099	Kein Adressraum mehr verfügbar

SAVE-VARIABLE-CONTAINER
Variablenbehälter sichern

Mit dem Kommando SAVE-VARIABLE-CONTAINER werden Variablenbehälter gesichert.

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**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

SELECT-VARIABLE-ELEMENTS
Elemente einer Listenvariablen auswählen

Das Kommando SELECT-VARIABLE-ELEMENTS gibt die Elemente einer S-Variablen (Listenvariablen) auf den Bildschirm aus und schreibt davon ausgewählte Elemente in eine andere 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>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler garantierte Meldungen: SDP1120, SPD1121, SPD1122
	64	SPD0259	Ausführung abgebrochen; Selektion ignoriert.
	130	SDP0099	Kein Adressraum mehr verfügbar

SEND-DATA
Datensatz an ein Programm übergeben

SEND-DATA sollte immer dann verwendet werden, wenn Datensätze und Kommandos gemischt werden sollen.
Das Auftreten eines SEND-DATA-Kommandos im "Datenstrom" löst nicht wie bei anderen Kommandos implizit eine EOF-Bedingung aus, sondern steuert dies über den Operanden RECORD.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

SEND-STMT
Anweisungssatz an ein Programm übergeben

SEND-STMT sollte immer dann verwendet werden, wenn Kommandos und Anweisungen gemischt werden. Ein SEND-STMT-Kommando im Datenstrom löst analog zu SEND-DATA keine implizite EOF-Bedingung aus.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

SET-PROCEDURE-OPTIONS
Prozedureigenschaften festlegen

Mit dem Kommando SET-PROCEDURE-OPTIONS kann der Benutzer die Eigenschaften einer *S-Prozedur* festlegen. Das Kommando ist optional. Wird es verwendet, so muss es das *erste* Kommando des Prozedurkopfs sein. Wird es nicht verwendet, sind die Eigenschaften gemäß den Voreinstellungen von SDF-P vereinbart.

SET-PROCEDURE-**OPT**IONS

CALLER = ***ANY** / ***CALL** / ***INCL**UDE
,**IMPL**ICIT-**DECL**ARATION = ***YES** / ***NO**
,**LOG**GING-**ALLOW**ED = ***PAR**AMETERS(...) / ***Y**ES / ***NO**
***PAR**AMETERS(...)
CMD = *YES / ***NO**
,**DATA** = *YES / ***NO**
,**INTERRUPT-ALLOW**ED = ***Y**ES / ***NO**
,**INP**UT-**FORM**AT = ***FREE-REC**ORD-**LENGTH** / ***BY-SDF-OPT**ION
,**DATA-ESC**APE-**CHAR** = ***NONE** / '&&' / '#' / '*' / '@' / '$' / ***STD**
,**SYS**TEM-**FILE-CONTEXT** = ***STD** / ***SAME-AS-CALL**ER / ***OWN**
,**DATA-ERROR**-HANDLING = ***Y**ES / ***NO**
,**JV-REPLACEMENT** = ***NONE** / ***AFTER-BUILTIN-FUNCTION**
,**ERROR-MECHANISM** = ***SPIN-OFF-COMPATIBLE** / ***BY-RETURNCODE**
,**SUP**PRESS-**SDP-MSG** = ***NONE** / list-poss(2000): <alphanum-name 7..7>

Kommando-Returncode

Die Kommando-Returncodes können nur auftreten, wenn das Kommando außerhalb des Prozedurkopf verwendet wird.

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

SET-VARIABLE
Variablen einen Wert zuweisen

Mit dem Kommando SET-VARIABLE werden einfachen oder zusammengesetzten Variablen Werte zugewiesen.

Bei Zuweisung von einfachen Variablen oder Variablenelementen müssen die Datentypen sich entsprechen. Bei globaler Zuweisung von zusammengesetzen Variablen muss auch die Struktur der Variablen übereinstimmen.

SET-VARIABLE Kurzname: **STV**

<composed-name$_1$ 1..255> = <text 0..1800 with-low *expr*> / <composed-name$_2$ 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**

Kommando-Returncode

Während der Zuweisung von Strukturen, Arrays oder Listen ist es möglich, dass ein Teil des Kommandos abgearbeitet und ausgeführt wurde, bevor ein Fehler auftrat. In diesem Fall ist das Resultat des Kommandos nicht garantiert.

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler garantierte Meldung: SDP1030
	130	SDP0099	Kein Adressraum mehr verfügbar

SHOW-STREAM-ASSIGNMENT
S-Variablenstrom anzeigen

SHOW-STREAM-ASSIGNMENT zeigt die aktuelle Zuweisung der angegebenen S-Variablenströme an.

SHOW-STREAM-ASSIGNMENT

STREAM-NAME = **<u>*ALL</u>** / ***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** = **<u>*SYSOUT</u>** / ***SYSLST**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	32	CMD2009	Fehler während S-Variablen-Ersetzung
	64	CMD0216	Erforderliches Privileg fehlt
	64	OPS0001	Kein Speicher für S-Variablen
	64	SDP0091	Semantikfehler
	64	SDP0517	Angegebener Variablenstrom-Name existiert nicht
	64	SDP0519	Kein Treffer für angegebene Wildcards

SHOW-STRUCTURE-LAYOUT
Elementnamen eines Strukturlayouts ausgeben

Das Kommando SHOW-STRUCTURE-LAYOUT gibt die Elementnamen eines Strukturlayouts aus. Das Strukturlayout muss zuvor mit BEGIN-STRUCTURE definiert worden sein.

SHOW-STRUCTURE-**LAYOUT** Kurzname: **SHSTRL**

NAME = ***ALL** / <structured-name 1..20 with-wild(40)> / list-poss(2000): <structured-name 1..20>
,**SCOPE** = ***VISIBLE** / ***PROC**EDURE / ***CURR**ENT / ***TASK**
,**OUTPUT** = ***SYSOUT** / ***SYSLST** / <filename 1..54 without-gen-vers>(...) / ***VAR**IABLE(...) / ***LIB**RARY-**ELEM**ENT(...)
<filename 1..54 without-gen-vers>(...)
WRITE-MODE = ***REPLACE** / ***EXTEND**
***VAR**IABLE(...)
VARIABLE-**NAME** = <composed-name 1..20>
,**WRITE-MODE** = ***REPLACE** / ***EXTEND**
***LIB**RARY-**ELEM**ENT(...)
LIBRARY = <filename 1..54 without-vers>
,**ELEM**ENT = <composed-name 1..64>(...)
<composed-name 1..64>(...)
VERSION = ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24>
,**TYPE** = **S** / <alphanum-name 1..8>

Kommando-Returncode

Es ist möglich, dass ein Teil des Kommandos schon abgearbeitet und ausgeführt wurde, bevor der Fehler auftrat. In diesem Fall ist das Ergebnis des Kommandos nicht garantiert.

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; kein Layout gefunden
2	0	SDP2000	Warnung: Nicht alle Elemente der Eingabeliste konnten erfolgreichabgearbeitet werden. garantierte Meldung: SDP2000
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	64	SDP1008	Variable existiert nicht
	64	SDP2001	Keines der Elemente konnte angezeigt werden
	130	SDP0099	Kein Adressraum mehr verfügbar

SHOW-VARIABLE
Inhalte von Variablen ausgeben

Gibt Inhalte von Variablen aus.
Wenn sich SHOW-VARIABLE auf eine Liste oder ein Array bezieht, werden die Elemente in der numerischen Folge ihrer Indizes ausgegeben.
Wenn sich SHOW-VARIABLE auf eine Struktur bezieht, werden die Elemente in der Reihenfolge der Elementdeklarationen ausgegeben.

SHOW-VARIABLE Kurzname: **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-ELEM**ENTS = **1** / ***REST** / <integer 1..2147483647>

,**SEL**ECT = ***BY-ATTR**IBUTES(...)

***BY-ATTR**IBUTES(...)**)**

SCOPE = ***VISIBLE** / ***PROC**EDURE / ***CURR**ENT / ***CURR**ENT-**PARAM**ETERS / ***TASK-VISIBLE** /
***TASK-ALL** / ***CALL**ING-**PROC**EDURES
,**INIT**IALIZATION = ***YES** / ***ANY**

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

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

VALUE = ***WITHOUT-QUOTES** / ***C-LIT**ERAL / ***X-LIT**ERAL / ***NONE**
,**NAME** = ***FULL-NAME** (...) / ***ELEM**ENT-**NAME** (...) / ***NONE**
***FULL-NAME(...)**
LIST-INDEX-NUMBER = ***NO** / ***Y**ES
***ELEM**ENT-**NAME(..)**
LIST-INDEX-NUMBER = ***NO** / ***Y**ES

,**OUTPUT** = ***SYSOUT** / ***SYSLST** / <filename 1..54 without-gen-vers>(...) / ***VAR**IABLE(...) /
***LIB**RARY-**ELEM**ENT(...)

<filename 1..54 without-gen-vers>(...)
WRITE-MODE = ***REPLACE** / ***EXTEND**

***VAR**IABLE(...)
VARIABLE-**NAME** = <composed-name 1..20>
,**WRITE-MODE** = ***REPLACE** / ***EXTEND**

***LIB**RARY-**ELEM**ENT(...)
LIBRARY = <filename 1..54 without-vers>
,**ELEM**ENT = <composed-name 1..64>(...)
<composed-name 1..64>(...)
VERSION = ***HIGH**EST-**EXIST**ING / ***UP**PER-**LIM**IT / <composed-name 1..24>
,**TYPE** = **S** / <alphanum-name 1..8>
,**WRITE-MODE** = ***REPLACE** / ***EXTEND**

Kommando-Returncode

Es ist möglich, dass ein Teil des Kommandos schon abgearbeitet und ausgeführt wurde, bevor der Fehler auftrat. In diesem Fall ist das Ergebnis des Kommandos nicht garantiert.

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Warnung; keine Variable gefunden
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler garantierte Meldung: SDP1008
	130	SDP0099	Kein Adressraum mehr verfügbar

SHOW-VARIABLE-ATTRIBUTES
Variablenattribute ausgeben

Das Kommando SHOW-VARIABLE-ATTRIBUTES informiert über die Attribute der angegebenen Variablen. Die Attribute umfassen Name der Variablen, Anfangswert, Datentyp, einfache oder zusammengesetzte Variable usw.

SHOW-VARIABLE-**ATTR**IBUTES

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-ELEM**ENTS = **1** / ***REST** / <integer 1..2147483647>

,**INF**ORMATION = ***NAME** / ***VAR**IABLE-**ATTR**IBUTES-ONLY / ***ALL-ATTR**IBUTES

,**ATT**ACHED-**INF**ORMATION = ***NO** / ***YES**

,**OUTPUT** = ***SYSOUT** / ***SYSLST**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler (Variable existiert nicht) garantierte Meldung: SDP1008
	130	SDP0099	Kein Adressraum mehr verfügbar

SHOW-VARIABLE-CONTAINER-ATTR
Offene Variablenbehälter anzeigen

Das Kommando SHOW-VARIABLE-CONTAINER-ATTR zeigt alle offenen Variablenbehälter an.

SHOW-VARIABLE-**CONTAINER-ATTR**

CONTAINER-NAME = ***ALL** / <composed-name 1..64 with-wild(80)> / list-poss: <composed-name 1..64>
,**CONTAINER-SCOPE** = ***VISIBLE** / ***PROC**EDURE / ***CURR**ENT / ***TASK**
,**OUTPUT** = ***SYSOUT** / ***SYSLST**

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

SORT-VARIABLE
Listenvariable sortieren

Das Kommando SORT-VARIABLE sortiert die Elemente einer Listenvariablen nach ihrem Inhalt in aufsteigender oder absteigender Reihenfolge. Es können nur Listenvariablen sortiert werden, deren Elemente einfache Variablen gleichen Typs (STRING, INTEGER, BOOLEAN oder ANY) sind.

SORT-VARIABLE

VARIABLE-**NAME** = list-poss(2000): <composed-name 1..255>
,**SORT**ING-**ORDER** = ***ASCEND**ING / ***DESCEND**ING

Kommando-Returncode

Wenn im Fehlerfall nur ein Teil des Kommandos abgearbeitet und ausgeführt wurde, ist das Resultat des Kommandos nicht garantiert.

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	SDP2000	Warnung; einige Elemente konnten nicht sortiert werden
	64	SDP2001	Keines der Elemente konnte sortiert werden

TRACE-PROCEDURE
Unterbrochene Prozedur schrittweise fortsetzen

Setzt eine unterbrochene Prozedur schrittweise fort. Die anschließenden Kommandos werden nur protokolliert, wenn Protokollierung zulässig ist.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

TRANSMIT-BY-STREAM
Variablen übertragen

Das Kommando TRANSMIT-BY-STREAM führt von Client-Seite eine Variablen-Übertragung über den angegebenen S-Variablenstrom zu oder vom angesprochenen Server durch. Wenn *DUMMY dem S-Variablenstrom zugewiesen ist, gibt das Kommando einen Returncode zurück, der besagt, dass nichts geändert wurde.

TRANSMIT-BY-STREAM

STREAM-NAME = <structured-name 1..20>

,**VAR**IABLE**-NAME** = ***NONE** / <composed-name 1..255>

,**RETURN-VAR**IABLE**-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>

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
1	0	CMD0001	Variablenstrom ist *DUMMY zugewiesen; keine Übertragung, Variable bleibt unverändert
2	0	SDP0512	Server ist nicht länger aktiv. Datenstrom ist *DUMMY zugewiesen
2	0	SDP0531	Warnung vom Server; Prozess wird fortgesetzt
	1	CMD0202	Syntaxfehler
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	CMD0216	Erforderliches Privileg fehlt
	64	SDP0091	Semantikfehler garantierte Meldungen: SDP1008
	64	SDP0532	Server-Fehler; Kommando abgewiesen
	64	SDP0534	Interner Server-Fehler; Kommando abgebrochen. Server-Verbindung abgebrochen nach unerwartetem Ereignis oder bei mangelhaften oder fehlenden System-Ressourcen
	64	SDP0517	Variablenstrom existiert nicht
	64	SDP0522	Übertragene Daten inkompatibel zum Format, das der Server bearbeitet
	64	SDP1132	Variablenname zu lang
	130	SDP0099	Kein Adressraum mehr verfügbar

UNTIL
REPEAT-Block abschließen

UNTIL enthält die Schleifenbedingung für einen REPEAT-Block, d. h. für eine REPEAT-Schleife, und schließt den REPEAT-Block ab (Einleitungskommando für den REPEAT-Block: REPEAT).

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	0	CMD0001	Ohne Fehler
	1	CMD0202	Syntaxfehler
	1	SDP0118	Kommando im falschen Kontext
	1	SDP0139	Back Branch-Grenze erreicht
	1	SDP0223	Falsche Umgebung
	3	CMD2203	Falsche Syntaxdatei
	32	CMD0221	Systemfehler (interner Fehler)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

WHILE
WHILE-Block einleiten

WHILE leitet einen WHILE-Block, das heißt eine WHILE-Schleife, ein. Die WHILE-Schleife muss mit dem Kommando END-WHILE abgeschlossen werden.

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

Kommando-Returncode

(SC2)	SC1	Maincode	Bedeutung
	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)
	64	SDP0091	Semantikfehler
	130	SDP0099	Kein Adressraum mehr verfügbar

Programmschnittstelle

Übersicht

Makros für Systemverwalter	Kurzbeschreibung
BIFDEF	Generiert den Tabelleneintrag, der den Namen der Funktion mit den Adressen des Ausführungmodul-Eintrags und des Syntaxbeschreibung-Eintrags verbindet
BIFDESC	Enthält die Syntaxbeschreibung von Systemverwalter-Funktionen
BIFMDL1	Generiert die DSECT für die Werte der Systemverwalter-Funktionen
BIFMDL2	Generiert den DSECT für den Returncode der Systemverwalter-Funktionen

Makros für Anwender	Kurzbeschreibung
CLIEXPR	SDF-P-Ausdrücke auswerten
CLIGET	Unterbrechungsschutz für eine Prozedur abfragen
CLISET	expliziten Schutz vor Programmunterbrechungen vereinbaren
CMD	SDF-P-Kommandos aus einem Programm absetzen
GETVAR	Lesen von einfachen und zusammengesetzten Variablen
PUTVAR	Schreiben von einfachen und zusammengesetzten Variablen
SHOWSSA	über Variablenstrom-Zuweisung informieren
TRANSVV	Variablen mittels Variablenstrom übertragen
VARINF	zusammengesetzte Variablen bearbeiten (modifizieren)

Makros

BIFDEF

Der Makro BIFDEF generiert den Tabelleneintrag, der den Namen der Funktion mit den Adressen des Ausführungmodul-Eintrags und des Syntaxbeschreibung-Eintrags verbindet.

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

BIFDESC

Der Makroaufruf BIFDESC enthält die Syntaxbeschreibung von Systemverwalter-Funktionen.

BIFDESC definiert eine statische Struktur, die ausschließlich vom Subsystem SDF-P-BIF benutzt wird.

Operation	Operanden
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

Der Makroaufruf BIFMDL1 generiert die DSECT für die Werte der Systemverwalter-Funktionen. Es wird die Struktur jedes Elements der Operandenliste für das Ausführungsmodul beschrieben.

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

BIFMDL2

Der Makroaufruf BIFMDL2 generiert die DSECT für den Returncode der Systemverwalter-Funktionen. Es wird der Returncode beschrieben, der vom Ausführungsmodul zurückgegeben wird.

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

CLIEXPR

Der Makro CLIEXPR wertet arithmetische, logische und String-Ausdrücke aus. Der Ausdruck wird in einem Input-Feld übergeben, das Ergebnis wird in ein Output-Feld zurückgeschrieben. Es kann spezifiziert werden, wie das Ergebnis zurückgeliefert werden soll (Dualzahl, boolsche Konstante, String).
Der Makro kann auch mit MF=M aufgerufen werden.

Operation	Operanden
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>

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Normale Ausführung
01	00	0000	Überlauf: PROT-Feld (Warnung)
00	40	0001	Syntaxfehler im auszuwertenden Ausdruck
01	40	0001	Überlauf PROT-Feld
00	40	0002	Fehler beim Auswerten des Ausdrucks
01	40	0002	Überlauf: PROT-Feld
00	40	0003	OUTPUT-Feld zu klein
00	01	0004	INPUT-Feld nicht spezifiziert oder nicht ausgerichtet
01	01	0004	OUTPUT-Feld nicht spezifiziert oder nicht ausgerichtet
02	01	0004	Protokoll-Feld (nicht ausgerichtet)
03	01	0004	andere Felder (nicht ausgerichtet)
04	01	0004	Feldadresse angegeben, aber kein Zugriff möglich
00	40	0005	Kein ausreichend freier Platz im Adressraum des Aufrufers
01	20	0006	Systemfehler
00	40	0007	Ungültiges Prozedurformat; Makroausführung abgebrochen
00	01	FFFF	Falsche Angabe für UNIT oder FUNCTION im Standard-header
00	02	FFFF	Die angeforderte Funktion wird nicht unterstützt
00	03	FFFF	Falsche Versionsangabe im Standardheader

CLIGET

Mit dem Makroaufruf CLIGET kann ein Programm abfragen, ob ein Schutz vor impliziten Unterbrechungen erforderlich ist.
CLIGET liefert die Einstellung des Operanden INTERRUPT-ALLOWED, die in den Kommandos SET-PROCEDURE-OPTIONS, MODIFY-PROCEDURE-OPTIONS oder BEGIN-PROCEDURE gesetzt wurde.

Operation	Operanden
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

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler; Parameter zu kurz
00	20	0004	Systemfehler
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste

CLISET

Der Makroaufruf CLISET schützt Programme explizit vor Unterbrechungen.

Operation	Operanden
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]

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
01	00	0000	Keine Aktion (EXNINT=*U)
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler;
01	00	0002	EXNINT=*Y wurde schon vorher gesetzt oder EXNINT=*N wurde schon vorher gesetzt
00	20	0004	Systemfehler
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste

CMD

Mit dem Makro CMD können in Assembler-Programmen Kommandos ausgeführt werden, auch SDF-P-Kommandos.

Kommando	**Funktion**
ASSIGN-STREAM	S-Variablenstrom zuweisen
BEGIN-STRUCTURE	Deklaration einer statischen Struktur beginnen
CALL-PROCEDURE	Prozedur starten
CLOSE-VARIABLE-CONTAINER	Variablenbehälter schließen
DECLARE-CONSTANT	Variable mit konstantem Wert definieren
DECLARE-ELEMENT	Strukturelement deklarieren
DECLARE-VARIABLE	Variable deklarieren
DELETE-STREAM	S-Variablenstrom löschen
DELETE-VARIABLE	Variable löschen
END-STRUCTURE	Ende einer Strukturdeklaration kennzeichnen
ENTER-PROCEDURE	Prozedur als Hintergrundprozedur starten
FREE-VARIABLE	Variableninhalt löschen
IMPORT-VARIABLE	Variable importieren
INCLUDE-CMD	Kommandofolgen übergeben
INCLUDE-PROCEDURE	INCLUDE-Prozedur starten
MODIFY-PROCEDURE-OPTIONS	Prozedureigenschaften ändern
OPEN-VARIABLE-CONTAINER	Variablenbehälter öffnen
READ-VARIABLE	Variablenwerte einlesen
SAVE-VARIABLE-CONTAINER	Variablenbehälter sichern
SELECT-VARIABLE-ELEMENTS	Elemente einer Listvariablen auswählen
SHOW-STREAM-ASSIGNMENT	Zuweisung für S-Variablenstrom ausgeben
SHOW-STRUCTURE-LAYOUT	Elementnamen eines Strukturlayouts ausgeben
SHOW-VARIABLE	Variableninhalte ausgeben
SHOW-VARIABLE-ATTRIBUTES	Variablenattribute ausgeben
SHOW-VARIABLE-CONTAINER-ATTR	Variablenbehälter-Attribute ausgeben
SORT-VARIABLE	Sortiert die Elemente einer Listenvariablen
TRANSMIT-BY-STREAM	Variablen mit S-Variablenstrom übertragen

GETVAR

Der Makroaufruf GETVAR liest den Inhalt einer Variablen.
GETVAR kann auf einfache Variablen und Elemente zusammengesetzter Variablen angewendet werden.

Operation	Operanden
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>

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler
00	01	0002	Syntaxfehler im Variablennamen
00	40	0003	Area zu klein
00	40	0004	Variable nicht deklariert
00	40	0005	Variablenbehälter nicht verfügbar
00	40	0006	Datentyp und Variablenwert nicht vereinbar
00	40	0008	Variable hat keinen Wert
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste

PUTVAR

Der Makroaufruf PUTVAR weist einer Variablen einen Wert zu. PUTVAR kann auf einfache Variablen und Elemente zusammengesetzter Variablen angewendet werden.
Wenn eine einfache Variable bei der Zuweisung noch nicht existiert, dann wird sie angelegt, falls IMPLICIT-DECLARATION=YES und IMPDEC = *STD gilt oder im Makroaufruf IMPDEC=*YES angegeben ist.

Operation	Operanden
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

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler
00	01	0002	Syntaxfehler im Variablennamen
00	40	0004	Variable nicht deklariert
00	40	0005	Variablenbehälter nicht verfügbar
00	40	0006	Datentyp und Variablenwert nicht vereinbar
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste

SHOWSSA

Der Makroaufruf SHOWSSA zeigt die aktuelle Zuweisung des angegebenen S-Variablenstroms an. SHOWSSA ist funktionsgleich mit dem Kommando SHOW-STREAM-ASSIGNMENT. Jedoch kann mit SHOWSSA keine Liste von Strömen angegeben werden.

Operation	Operanden
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

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler
00	40	0002	Angegebener Variablenstrom unvollständig
00	40	0003	SSTA-Fehler (SSTA zu klein, nicht initialisiert, ...)
00	40	0004	Abgebrochen durch K2 während Ausgabe auf SYSOUT
00	40	0005	Fehler während Ausgabe auf SYSOUT
00	40	0006	Fehler während Ausgabe auf SYSLST
02	01	0007	Mehr als ein Variablenstrom für OUTPUT= *RETURNCODE
00	20	0008	System-Fehler
02	00	000A	Angegebener Variablenstrom ist *DUMMY zugewiesen
02	00	000B	Angegebener Variablenstrom ist schon zugewiesen
02	00	000C	Angegebener Variablenstrom existiert nicht
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste
00	41	FFFF	SDF-P ist nicht geladen
00	81	FFFF	SDF-P arbeitet zwischenzeitlich nicht

TRANSVV

Der Makroaufruf TRANSVV führt von Client-Seite eine Variablenübertragung über den angegebenen S-Variablenstrom durch, zu dem aktuell zugewiesenen Server (Kommando ASSIGN-STREAM). TRANSVV ist funktionsgleich mit dem Kommando TRANSMIT-BY-STREAM. Es können nur S-Variablenströme angegeben werden, die auf der gleichen Prozedurstufe zugewiesen wurden, auf der das Programm gestartet wurde.

Operation	**Operanden**
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]

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Übertragung erfolgreich beendet; kein Fehler
01	00	0000	Variablenstrom wurde *DUMMY zugewiesen; keine Übertragung
00	01	0001	Parameter-Fehler
00	40	0002	Angegebener Variablenstrom unvollständig
00	40	0003	Angegebene Variable unvollständig
00	40	0004	RET-SSTA zu klein (reserviert für Software-Entwickler)
00	01	0005	Übertragene Daten (Benutzer- oder Kontrolldaten) sind nicht mit dem Format, das der Server verarbeiten kann, kompatibel
00	40	0006	Fehlermeldung vom Server; gespeichert in RCNAME (wenn angegeben)
02	00	0007	Warnung vom Server; gespeichert in RCNAME (wenn angegeben)
02	00	0008	Variablenstrom auf *DUMMY zurückgesetzt; Server ist nicht mehr aktiv
00	20	0009	System-Fehler
00	20	000A	Fehler bei der Server-Verbindung
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste
00	41	FFFF	SDF-P ist nicht geladen
00	81	FFFF	SDF-P arbeitet zwischenzeitlich nicht

VARINF

Mit dem Makroaufruf VARINF können Variablen analysiert werden, auch zusammengesetzte Variablen, deren Elemente selbst wieder zusammengesetzte Variablen sind.

Operation	Operanden
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>

Returncodes (hexadezimal)

Subcode2	Subcode1	Maincode	Bedeutung
00	00	0000	Makroaufruf war erfolgreich; kein Fehler
00	01	0001	Parameter-Fehler
00	01	0002	Syntaxfehler im Variablennamen
00	40	0003	Area zu klein
00	40	0004	Variable nicht deklariert
00	40	0005	Variablenbehälter nicht verfügbar
00	40	0007	Letztes Variablenelement erreicht, kein weiteres Variablenelement vorhanden
00	01	FFFF	Unbekannte Unit- oder Funktions-Nummer
00	02	FFFF	Funktion nicht verfügbar
00	03	FFFF	Falsche Version der Operandenliste

Bildung von Variablennamen

Für Variablennamen und Variablenteilnamen in SDF-P werden in der Kommandosyntax die SDF-Datentypen <composed-name> und <structured-name> verwendet. Die tatsächlich von SDF-P überprüfte Syntax ist aber gegenüber diesen Datentypen eingeschränkt und soll deshalb hier näher beschrieben werden.

Ein Variablenname (composed-variable-name) in SDF-P setzt sich zusammen aus einem oder mehreren Strukturnamen (structured-variable-name), die durch Punkte voneinander getrennt sind. Durch den Punkt wird jeweils der Name einer Unterstruktur gekennzeichnet. Leerzeichen innerhalb eines Variablennamens sind nicht zugelassen.

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

Länge	mindestens 1 Zeichen, höchstens 255 Zeichen

Ein Strukturname besteht aus einem Variablenteilnamen (vtn), der von einem # und einem Indexnamen gefolgt sein kann, falls er ein Listenelement oder ein Arrayelement bezeichnet.

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

Länge	mindestens 1 Zeichen, höchstens 20 Zeichen
Variablenteilname	(vtn):
Zeichenvorrat	alle Buchstaben (A, ... Z) alle Ziffern (0, ... 9) @ und Bindestrich (-)
Erstes Zeichen	Buchstabe
Konventionen	Zeichenfolge SYS am Beginn des Variablennamens ist reserviert für Systemvariablen und sollte vom Benutzer nicht verwendet werden. Es dürfen nicht zwei Bindestriche hintereinander stehen. Der Bindestrich darf nicht das letzte Zeichen des Variablenteilnamens sein.

Falls der Strukturname ein Listenelement oder ein Arrayelement bezeichnet, so folgt ihm ein # mit einem Indexnamen.

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

Elementnamen

Bei Elementnamen muss unterschieden werden zwischen Arrayelementen, Listenelementen und Strukturelementen.
Generell gilt für Elementnamen die Syntax, wie unter <composed-variable-name> beschrieben.

Listenelementnamen

Listenelementnamen bestehen an der Benutzerschnittstelle aus folgenden Komponenten:

– Name der Liste (<composed-variable-name 1..253>)
– #
– optional: Elementnummer (<integer 1..2147483647>)

Arrayelementnamen

Arrayelementnamen setzen sich zusammen aus folgenden Komponenten:

– Name des Arrays (<composed-variable-name 1..253>)
– # (Kennzeichen für Arrayelemente, wenn weitere Zeichen folgen)
– Arrayindex (<integer -2147483648..2147483647>)

Der Listenindex sowie der Arrayindex können auch über eine einfache Variable bezeichnet werden, die einen Integer-Wert im gültigen Wertebereich enthält.

SDF-P analysiert die Zeichenfolge, die im Elementnamen auf das Aggregatsymbol # folgt: Ist das erste Zeichen eine Ziffer oder ein Bindestrich (= Minuszeichen), wird die Zeichenfolge als Integer-Wert interpretiert, also als direkte Angabe des Index.

Ist das erste Zeichen ein Buchstabe, wird die Zeichenfolge als Variablenname interpretiert. SDF-P sucht dann eine Variable mit dem angegeben Namen. Dieser Name muss eine einfache Variable bezeichnen, die mit einem gültigen Integer-Wert initialisiert sein muss. Ist das nicht der Fall, wird eine Fehlermeldung ausgegeben. Wenn die Variable einen gültigen Integer-Wert enthält, wird dieser Wert im Arrayindex bzw. im Listenindex eingesetzt und auf das so bezeichnete Aggregatelement zugegriffen.

Strukturelementnamen

Strukturelementnamen setzen sich aus folgenden Komponenten zusammen:

– Name der Struktur (<composed-variable-name 1..253>)
– "." (als Kennzeichen für Strukturen)
– Teilname des Elements (<structured-variable-name 1..20>)

Reservierte Wörter

"Reservierte Wörter" sind Schlüsselwörter, die für Operatoren in Ausdrücken oder boolesche Konstanten verwendet werden. Sie dürfen nicht als Variablennamen deklariert werden.

Wenn sie dennoch in einer Variablen-Deklaration eingesetzt werden, wird keine Variable angelegt, es wird eine Fehlermeldung ausgegeben.

Folgende Schlüsselwörter sind reservierte Wörter:

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

Reservierte Variablennamen

Variablennamen, die mit dem Präfix SYS beginnen sind reserviert für den Datentransfer von und zu Systemkomponenten. Dabei bildet der Variablenname SYSSWITCH eine besondere Form.

SYSSWITCH

Der Variablenname SYSSWITCH bezeichnet eine zusammengesetzte Variable vom Typ Array, über die die Auftragsschalter angesprochen werden können.

Das Array SYSSWITCH ist folgendermaßen definiert:

Datentyp	BOOLEAN
Geltungsbereich	TASK (in Dialog- und Prozedurumgebung)
Anzahl Elemente	32
Arrayindex	0,..., 31
Werte	TRUE = Schalter steht auf ON, FALSE = Schalter steht auf OFF

Die Arrayelemente können über die Teilnamen SYSSWITCH#0 bis SYSSWITCH#31 angesprochen werden. Es ist sowohl Schreib- als auch Lesemodus zulässig.

Das Array SYSSWITCH oder die Arrayelemente können nicht gelöscht werden, das heißt, sie können nicht in einem FREE-VARIABLE- bzw. DELETE-VARIABLE-Kommando angegeben werden.

Das Array SYSSWITCH ist immer implizit in jeder Prozedur deklariert.

SYSPARAM

Der Variablenname SYSPARAM bezeichnet eine Variable vom Typ String, über die auf die mit den Kommandos START-/LOAD-EXECUTABLE-PROGRAM übergebenen Programmparameter zugegriffen werden kann. In C-Programmen erfolgt der Zugriff auf die Programmparameter mit der Funktion `getopt`; Assemblerprogramme müssen die Variable SYSPARAM über den Makroaufruf GETVAR (siehe Seite 118) einlesen und selbst auswerten.

Metasyntax, Datentypen und Operatoren

Metasyntax für Kommandos und Makros

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter (Kommando-, Anweisungs-, Operandennamen, Schlüsselwortwerte) und konstante Operandenwerte. Schlüsselwortwerte beginnen mit *.	**HELP-SDF** **SCREEN-STEPS** = ***NO**
GROSSBUCHSTABEN in Halbfett	Großbuchstaben in Halbfett kennzeichnen garantierte bzw. vorgeschlagene Abkürzungen der Schlüsselwörter.	**GUID**ANCE**-MODE** = ***Y**ES
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörenden Operandenwerten.	**GUID**ANCE**-MODE** = ***NO**
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird.	**SYNTAX-F**ILE = <filename 1..54>
Unterstreichung	Der Unterstrich kennzeichnet den Default-Wert eines Operanden.	**GUID**ANCE**-MODE** = ***NO**
/	Der Schrägstrich trennt alternative Operandenwerte.	**NEXT-FIELD** = ***NO** / ***Y**ES
(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	,**UNGUID**ED**-DIA**LOG = ***Y**ES (...) / ***NO**
[ ]	Eckige Klammern kennzeichnen struktureinleitende Operandenwerte, deren Angabe optional ist. Die nachfolgende Struktur kann ohne den einleitenden Operandenwert angegeben werden.	**SELECT** = [***BY-ATTR**IBUTES](...)
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	,**GUID**ED**-DIA**LOG = ***Y**ES (...) ***Y**ES(...) **SCREEN-STEPS** = ***NO** / ***Y**ES

Kennzeichnung	Bedeutung	Beispiele
|	Der Strich kennzeichnet zusammengehörende Operanden einer Struktur. Sein Verlauf zeigt Anfang und Ende einer Struktur an. Innerhalb einer Struktur können weitere Strukturen auftreten. Die Anzahl senkrechter Striche vor einem Operanden entspricht der Strukturtiefe.	**SUP**PORT = ***TAPE**(...) ***TAPE(...)** | **VOL**UME = ***ANY**(...) | ***ANY**(...) | | ...
,	Das Komma steht vor weiteren Operanden der gleichen Strukturstufe.	**GUID**ANCE-**MODE** = ***NO** / ***Y**ES ,**SDF-COM**MANDS = ***NO** / ***Y**ES
list-poss(n):	Aus den list-poss folgenden Operandenwerten kann eine Liste gebildet werden. Ist (n) angegeben, können maximal n Elemente in der Liste vorkommen. Enthält die Liste mehr als ein Element, muss sie in runde Klammern eingeschlossen werden.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <filename 1..54>
Kurzname:	Der darauf folgende Name ist ein garantierter Aliasname des Kommando- bzw. Anweisungsnamens.	**HELP-SDF** Kurzname: **HPSDF**

Metasyntax für Funktionen

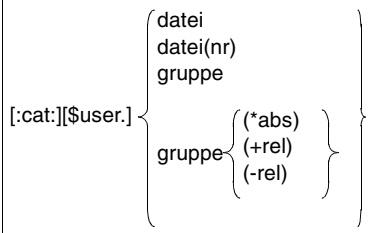
Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Bezeichnen Schlüsselwörter; einige Schlüsselwörter beginnen mit einem *	POSITION = *FIRST
=	Verbindet einen Operandennamen mit seinen Operandenwerten.	STREAM = *COMMAND / *DATA
	Trennt alternative Operandenwerte	STREAM = *COMMAND / *DATA
Unterstreichung	Kennzeichnet den Standardwert eines Operanden.	DIRECTION = ***FORWARD** / RESERVE
ausdruck	beliebiger Ausdruck	**EXPRESSION = ausdruck**
string_ausdruck	Ausdruck, der einen String liefert	STRING = string_ausdruck
arithm_ausdruck	Ausdruck, der einen numerischen Wert liefert	INTEGER = arithm_ausdruck
zahl	Ganzzahl (Integer-Wert)	START = 1 / zahl
zeichen	Zeichen im EBCDI-Code	FILL-BYTE = zeichen
msg_id	siebenstelliger Meldungsschlüssel	MSD-IDENTIFICATION = msg_id

Metasyntax für Variablennamen

Kennzeichnung	Bedeutung	Beispiele
:=	Definition	varname:=
<>	Datentyp (siehe Abschnitt „Datentypen“)	<composed-name 1...255>
m..n	Wertebereich (im Datentyp) Wiederholung	1..255
[ ]	optionale Angabe	[<symbol>]
/	Alternative Angaben	

Datentypen

Datentyp	Zeichenvorrat	Besonderheiten
alphanum-name	A…Z 0…9 $, #, @	
cat-id	A…Z 0…9	maximal 4 Zeichen; darf nicht mit der Zeichenfolge PUB beginnen
command-rest	beliebig	
composed-name	A…Z 0…9 $, #, @ Bindestrich Punkt Katalogkennung	alphanumerische Zeichenfolge, die in mehrere durch Punkt oder Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann. Ist auch die Angabe eines Dateinamens möglich, so kann die Zeichenfolge mit einer Katalogkennung im Format :cat: beginnen (siehe Datentyp filename).
c-string	EBCDIC-Zeichen	ist in Hochkommata einzuschließen; der Buchstabe C kann vorangestellt werden; Hochkommata innerhalb des c-string müssen verdoppelt werden
date	0…9 Strukturkennzeichen: Bindestrich	Eingabeformat: jjjj-mm-tt jjjj: Jahr; wahlweise 2- oder 4-stellig mm: Monat tt: Tag
device	A…Z 0…9 Bindestrich	Zeichenfolge, die maximal 8 Zeichen lang ist und einem im System verfügbaren Gerät entspricht. In der Dialogführung zeigt SDF die zulässigen Operandenwerte an. Hinweise zu möglichen Geräten sind der jeweiligen Operandenbeschreibung zu entnehmen.
fixed	+, - 0…9 Punkt	Eingabeformat: [zeichen][ziffern].[ziffern] [zeichen]: + oder - [ziffern]: 0...9 muss mindestens eine Ziffer, darf aber außer dem Vorzeichen maximal 10 Zeichen (0…9, Punkt) enthalten

Datentyp	Zeichenvorrat	Besonderheiten
filename	A…Z 0…9 $, #, @ Bindestrich Punkt	Eingabeformat:  :cat: wahlfreie Angabe der Katalogkennung; Zeichenvorrat auf A…Z und 0…9 eingeschränkt; max. 4 Zeichen; ist in Doppelpunkte einzuschließen; voreingestellt ist die Katalogkennung, die der Benutzerkennung laut Eintrag im Benutzerkatalog zugeordnet ist. $user. wahlfreie Angabe der Benutzerkennung; Zeichenvorrat ist A…Z, 0…9, $, #, @; max. 8 Zeichen; darf nicht mit einer Ziffer beginnen; $ und Punkt müssen angegeben werden; voreingestellt ist die eigene Benutzerkennung. $. (Sonderfall) System-Standardkennung datei Datei- oder Jobvariablenname; kann durch Punkt in mehrere Teilnamen gegliedert sein: $name_1$[.$name_2$[...]] $name_i$ enthält keinen Punkt und darf nicht mit Bindestrich beginnen oder enden; datei ist max. 41 Zeichen lang, darf nicht mit $ beginnen und muss mindestens ein Zeichen aus A…Z enthalten. #datei (Sonderfall) @datei (Sonderfall) # oder @ als erstes Zeichen kennzeichnet je nach Systemparameter temporäre Dateien und Jobvariablen.

Datentyp	Zeichenvorrat	Besonderheiten
filename (Forts.)		datei(nr) Banddateiname nr: Versionsnummer; Zeichenvorrat ist A…Z, 0…9, $, #, @. Klammern müssen angegeben werden. gruppe Name einer Dateigenerationsgruppe (Zeichenvorrat siehe unter "datei") gruppe { (*abs) / (+rel) / (-rel) } (*abs) absolute Generationsnummer (1..9999); * und Klammern müssen angegeben werden. (+rel) (-rel) relative Generationsnummer (0..99); Vorzeichen und Klammern müssen angegeben werden.
integer	0…9, +, -	+ bzw. - kann nur erstes Zeichen (Vorzeichen) sein.
name	A…Z 0…9 $, #, @	darf nicht mit einer Ziffer beginnen.
partial-filename	A…Z 0…9 $, #, @ Bindestrich Punkt	Eingabeformat: [:cat:][$user.][teilname.] :cat: siehe filename $user. siehe filename teilname wahlfreie Angabe des gemeinsamen ersten Namensteils von Dateien und Dateigenerationsgruppen in der Form: $name_1$.[$name_2$.[...]] $name_i$ siehe filename. Das letzte Zeichen von teilname muss ein Punkt sein. Es muss mindestens einer der Teile :cat:, $user. oder teilname angegeben werden.

Datentyp	Zeichenvorrat	Besonderheiten
posix-filename	A...Z 0...9 Sonderzeichen	Zeichenfolge, die maximal 255 Zeichen lang ist. Besteht entweder aus einem oder zwei Punkten, oder aus alphanumerischen Zeichen und Sonderzeichen; Sonderzeichen sind mit dem Zeichen \ zu entwerten. Nicht erlaubt ist das Zeichen /. Muss in Hochkommata eingeschlossen werden, wenn alternative Datentypen zulässig sind, Separatoren verwendet werden oder das erste Zeichen ?, ! bzw. ^ ist. Zwischen Groß- und Kleinschreibung wird unterschieden.
posix-pathname	A...Z 0...9 Sonderzeichen Strukturkennzeichen: Schrägstrich	Eingabeformat: [/]$part_1$[/.../$part_n$] wobei $part_i$ ein posix-filename ist; maximal 1023 Zeichen; muss in Hochkommata eingeschlossen werden, wenn alternative Datentypen zulässig sind, Separatoren verwendet werden oder das erste Zeichen ?, ! bzw. ^ ist.
product-version	A…Z 0…9 Punkt Hochkomma	Eingabeformat: [[C]'][V][m]m.naso['] Korrekturstand Freigabestand wobei m, n, s und o jeweils eine Ziffer und a ein Buchstabe ist. Ob Freigabe- und/oder Korrekturstand angegeben werden dürfen oder ob sie angegeben werden müssen, bestimmen Zusätze zu dem Datentyp (siehe Zusätze without-corr, without-man, mandatory-man und mandatory-corr). product-version kann in Hochkommata eingeschlossen werden, wobei der Buchstabe C vorangestellt werden kann. Die Versionsangabe kann mit dem Buchstaben V beginnen.
structured-name	A…Z 0…9 $, #, @ Bindestrich	alphanumerische Zeichenfolge, die in mehrere durch Bindestrich getrennte Teilzeichenfolgen gegliedert sein kann; erstes Zeichen: A…Z oder $, #, @
text	beliebig	Das Eingabeformat ist den jeweiligen Operandenbeschreibungen zu entnehmen.

Datentyp	Zeichenvorrat	Besonderheiten
time	0…9 Strukturkennzeichen: Doppelpunkt	Angabe einer Tageszeit Eingabeformat: { hh:mm:ss | hh:mm | hh } hh: Stunden mm: Minuten ss: Sekunden Führende Nullen können weggelassen werden
vsn	a) A…Z 0…9 b) A…Z 0…9 $, #, @	a) Eingabeformat: pvsid.folgenummer max. 6 Zeichen; pvsid: 2-4 Zeichen; Eingabe von PUB nicht erlaubt folgenummer: 1-3 Zeichen b) max. 6 Zeichen; PUB darf vorangestellt werden, dann dürfen jedoch nicht $, #, @ folgen.
x-string	Sedezimal: 00…FF	ist in Hochkommata einzuschließen; der Buchstabe X muss vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.
x-text	Sedezimal: 00…FF	ist nicht in Hochkommata einzuschließen; der Buchstabe X darf nicht vorangestellt werden; die Anzahl der Zeichen darf ungerade sein.

Zusätze zu Datentypen

Zusatz	Bedeutung
x..y *unit*	beim Datentyp integer: Intervallangabe x Mindestwert, der für integer erlaubt ist. x ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf. y Maximalwert, der für integer erlaubt ist. y ist eine ganze Zahl, die mit einem Vorzeichen versehen werden darf. *unit* Dimension. Folgende Angaben werden verwendet: *days* *byte* *hours* *2Kbyte* *minutes* *4Kbyte* *seconds* *Mbyte* *milliseconds*
x..y *special*	bei den übrigen Datentypen: Längenangabe Bei den Datentypen catid, date, device, product-version, time und vsn wird die Längenangabe nicht angezeigt. x Mindestlänge für den Operandenwert; x ist eine ganze Zahl. y Maximallänge für den Operandenwert; y ist eine ganze Zahl. x=y Der Operandenwert muss genau die Länge x haben. *special* Zusatzangabe zur Beschreibung eines Sonderdatentyps, der durch die Implementierung geprüft wird. Vor *special* können weitere Zusätze stehen. Folgende Angaben werden verwendet: *arithm-expr* arithmetischer Ausdruck (SDF-P) *bool-expr* logischer Ausdruck (SDF-P) *string-expr* String-Ausdruck (SDF-P) *expr* beliebiger Ausdruck (SDF-P) *cond-expr* bedingter Ausdruck (JV)
with	Erweitert die Angabemöglichkeiten für einen Datentyp.
-compl	Bei Angaben zu dem Datentyp date ergänzt SDF zweistellige Jahresangaben der Form jj-mm-tt zu: 20jj-mm-tt falls jj < 60 19jj-mm-tt falls jj ≥ 60
-low	Groß- und Kleinschreibung wird unterschieden.
-path-compl	Bei Angaben zu dem Datentyp filename ergänzt SDF die Katalog- und/oder die Benutzerkennung, falls diese nicht angegeben werden.
-under	Erlaubt Unterstriche '_' bei den Datentypen name und composed-name.

<table>
<tr><th>Zusatz</th><th>Bedeutung</th></tr>
<tr><td>with (Forts.)

-wild(n)</td><td>Teile eines Namens dürfen durch die folgenden Platzhalter ersetzt werden. n bezeichnet die maximale Eingabelänge bei Verwendung von Platzhaltern. Mit Einführung der Datentypen posix-filename und posix-pathname akzeptiert SDF neben den bisher im BS2000 üblichen Platzhaltern auch Platzhalter aus der UNIX-Welt (nachfolgend POSIX-Platzhalter genannt). Da derzeit nicht alle Kommandos POSIX-Platzhalter unterstützen, kann ihre Verwendung bei Datentypen ungleich posix-filename und posix-pathname zu Semantikfehlern führen.
Innerhalb einer Musterzeichenfolge sollten entweder nur BS2000- oder nur POSIX-Platzhalter verwendet werden. Bei den Datentypen posix-filename und posix-pathname sind nur POSIX-Platzhalter erlaubt. Ist eine Muster-zeichenfolge mehrdeutig auf einen String abbildbar, gilt der erste Treffer.
<table>
<tr><th>BS2000-Platzhalter</th><th>Bedeutung</th></tr>
<tr><td>*</td><td>Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.</td></tr>
<tr><td>Punkt am Ende</td><td>Teilqualifizierte Angabe eines Namens.
Entspricht implizit der Zeichenfolge "./*", d.h. nach dem Punkt folgt mindestens ein beliebiges Zeichen.</td></tr>
<tr><td>/</td><td>Ersetzt genau ein beliebiges Zeichen.</td></tr>
<tr><td><s_x:s_y></td><td>Ersetzt eine Zeichenfolge, für die gilt:
– sie ist mindestens so lang wie die kürzeste Zeichenfolge (s_x oder s_y)
– sie ist höchstens so lang wie die längste Zeichenfolge (s_x oder s_y)
– sie liegt in der alphabetischen Sortierung zwischen s_x und s_y; Zahlen werden hinter Buchstaben sortiert (A…Z, 0…9)
– s_x darf auch die leere Zeichenfolge sein, die in der alphabetischen Sortierung an erster Stelle steht
– s_y darf auch die leere Zeichenfolge sein, die an dieser Stelle für die Zeichenfolge mit der höchst möglichen Codierung steht (enthält nur die Zeichen X'FF')</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Zusatz</th><th colspan="2">Bedeutung</th></tr>
<tr><td rowspan="11">with-wild(n)
(Forts.)</td><td><s_1,…></td><td>Ersetzt alle Zeichenfolgen, auf die eine der mit s angegebenen Zeichenkombinationen zutrifft. s kann auch die leere Zeichenfolge sein. Jede Zeichenfolge s kann auch eine Bereichsangabe "s_x:s_y" sein (siehe Seite 138).</td></tr>
<tr><td>-s</td><td>Ersetzt alle Zeichenfolgen, die der angegebenen Zeichenfolge s nicht entsprechen. Das Minuszeichen darf nur am Beginn der Zeichenfolge stehen. Innerhalb der Datentypen filename bzw. partial-filename kann die negierte Zeichenfolge -s genau einmal verwendet werden, d.h., -s kann einen der drei Namensteile cat, user oder datei ersetzen.</td></tr>
<tr><td colspan="2">Platzhalter sind in Generations- und Versionsangaben von Dateinamen nicht erlaubt. In Benutzerkennungen ist die Angabe von Platzhaltern der Systemverwaltung vorbehalten. Platzhalter können nicht die Begrenzer der Namensteile cat (Doppelpunkte) und user ($ und Punkt) ersetzen.</td></tr>
<tr><td>POSIX-Platzhalter</td><td>Bedeutung</td></tr>
<tr><td>*</td><td>Ersetzt eine beliebige, auch leere Zeichenfolge. Ein * an erster Stelle muss verdoppelt werden, sofern dem * weitere Zeichen folgen und die eingegebene Zeichenfolge nicht mindestens einen weiteren Platzhalter enthält.</td></tr>
<tr><td>?</td><td>Ersetzt genau ein beliebiges Zeichen. Ist als erstes Zeichen außerhalb von Hochkommata nicht zulässig.</td></tr>
<tr><td>[c_x-c_y]</td><td>Ersetzt genau ein Zeichen aus dem Bereich c_x und c_y einschließlich der Bereichsgrenzen. c_x und c_y müssen einfache Zeichen sein.</td></tr>
<tr><td>[s]</td><td>Ersetzt genau ein Zeichen aus der Zeichenfolge s.
Die Ausdrücke [c_x-c_y] und [s] können kombiniert werden zu [s_1c_x-c_ys_2]</td></tr>
<tr><td>[!c_x-c_y]</td><td>Ersetzt genau ein Zeichen, das nicht in dem Bereich c_x und c_y einschließlich der Bereichsgrenzen enthalten ist. c_x und c_y müssen einfache Zeichen sein.
Die Ausdrücke [!c_x-c_y] und [!s] können kombiniert werden zu [!s_1c_x-c_ys_2]</td></tr>
<tr><td>[!s]</td><td>Ersetzt genau ein Zeichen, das nicht in der Zeichenfolge s enthalten ist. Die Ausdrücke [!s] und [!c_x-c_y] können kombiniert werden zu [!s_1c_x-c_ys_2]</td></tr>
</table>

<table>
<tr><th>Zusatz</th><th>Bedeutung</th></tr>
<tr><td>with (Forts.)

-wild-constr(n)</td><td>Angabe einer Konstruktionszeichenfolge, die angibt, wie aus einer zuvor angegebenen Auswahlzeichenfolge mit Musterzeichen (siehe with-wild) neue Namen zu bilden sind. n bezeichnet die maximale Eingabelänge bei Verwendung von Platzhaltern.
Die Konstruktionszeichenfolge kann aus konstanten Zeichenfolgen und Musterzeichen bestehen. Ein Musterzeichen wird durch diejenige Zeichenfolge ersetzt, die durch das entsprechende Musterzeichen in der Auswahlzeichenfolge ausgewählt wird.
Folgende Platzhalter können zur Konstruktionsangabe verwendet werden:
<table>
<tr><th>Platzhalter</th><th>Bedeutung</th></tr>
<tr><td>*</td><td>Entspricht der Zeichenfolge, die durch den Platzhalter * in der Auswahlzeichenfolge ausgewählt wird.</td></tr>
<tr><td>Punkt am Ende</td><td>Entspricht der teilqualifizierten Angabe eines Namens in der Auswahlzeichenfolge.
Entspricht der Zeichenfolge, die durch den Punkt am Ende der Auswahlzeichenfolge ausgewählt wird.</td></tr>
<tr><td>/ oder ?</td><td>Entspricht dem Zeichen, das durch den Platzhalter / oder ? in der Auswahlzeichenfolge ausgewählt wird.</td></tr>
<tr><td><n></td><td>Entspricht der Zeichenfolge, die durch den n-ten Platzhalter in der Auswahlzeichenfolge ausgewählt wird; n = <integer></td></tr>
</table>
Zuordnung der Platzhalter zu entsprechenden Platzhaltern in der Auswahlzeichenfolge:
In der Auswahlzeichenfolge werden alle Platzhalter von links nach rechts aufsteigend nummeriert (globaler Index).
Gleiche Platzhalter in der Auswahlzeichenfolge werden zusätzlich von links nach rechts aufsteigend nummeriert (platzhalter-spezifischer Index).
In der Konstruktionsangabe können Platzhalter auf zwei, sich gegenseitig ausschließende Arten angegeben werden:
1. Platzhalter werden über den globalen Index angegeben: <n>
2. Angabe desselben Platzhalters, wobei die Ersetzung gemäß dem platzhalter-spezifischen Index entsprechend erfolgt:
z.B. der zweite "/" entspricht der Zeichenfolge, die durch den zweiten "/" in der Auswahlzeichenfolge ausgewählt wird.</td></tr>
</table>

Zusatz	Bedeutung
with-wild-constr(n) (Forts.)	Bei Konstruktionsangaben sind folgende Regeln zu beachten: – Die Konstruktionsangabe kann nur Platzhalter der Auswahlzeichenfolge enthalten. – Soll die Zeichenkette, die der Platzhalter <...> bzw. [...] auswählt, in der Konstruktionsangabe verwendet werden, muss die Index-Schreibweise gewählt werden. – Die Index-Schreibweise muss gewählt werden, wenn die Zeichenkette, die einen Platzhalter der Auswahlzeichenfolge bezeichnet, in der Konstruktionsangabe mehrfach verwendet werden soll: Bei der Auswahlangabe "A/" muss z.B. statt "A//" die Konstruktionszeichenfolge "A<n><n>" angegeben werden. – Der Platzhalter * kann auch die leere Zeichenkette sein. Insbesondere ist zu beachten, dass bei mehreren Sternen in Folge (auch mit weiteren Platzhaltern) nur der letzte Stern eine nicht leere Zeichenfolge sein kann: z.B. bei "****" oder "*//*". – Aus der Konstruktionsangabe sollten gültige Namen entstehen. Darauf ist sowohl bei der Auswahlangabe als auch bei der Konstruktionsangabe zu achten. – Abhängig von der Konstruktionsangabe können aus unterschiedlichen Namen, die in der Auswahlangabe ausgewählt werden, identische Namen gebildet werden: z.B. "A/*" wählt die Namen "A1" und "A2" aus; die Konstruktionsangabe "B*" erzeugt für beide Namen denselben neuen Namen "B". Um dies zu vermeiden, sollten in der Konstruktionsangabe alle Platzhalter der Auswahlangabe mindestens einmal verwendet werden. – Wird die Konstruktionsangabe mit einem Punkt abgeschlossen, so muss auch die Auswahlzeichenfolge mit einem Punkt enden. Die Zeichenfolge, die durch den Punkt am Ende der Auswahlzeichenfolge ausgewählt wird, kann in der Konstruktionsangabe nicht über den globalen Index angegeben werden.

<table>
<tr><th>Zusatz</th><th>Bedeutung</th></tr>
<tr><td>with-wild-constr(n) (Forts.)</td><td>Beispiele:
<table>
<tr><th>Auswahlmuster</th><th>Auswahl</th><th>Konstruktionsmuster</th><th>neuer Name</th></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) Punkt am Ende des Namens kann Namenskonvention widersprechen (z.B bei vollqualifizierten Dateinamen)</td></tr>
<tr><td>without</td><td>Schränkt die Angabemöglichkeiten für einen Datentyp ein.</td></tr>
<tr><td>-cat</td><td>Die Angabe einer Katalogkennung ist nicht erlaubt.</td></tr>
<tr><td>-corr</td><td>Eingabeformat: [[C]']['V][m]m.na[']
Angaben zum Datentyp product-version dürfen den Korrekturstand nicht enthalten.</td></tr>
<tr><td>-gen</td><td>Die Angabe einer Dateigeneration oder Dateigenerationsgruppe ist nicht erlaubt.</td></tr>
<tr><td>-man</td><td>Eingabeformat: [[C]']['V][m]m.n[']
Angaben zum Datentyp product-version dürfen weder Freigabe- noch Korrekturstand enthalten.</td></tr>
<tr><td>-odd</td><td>Der Datentyp x-text erlaubt nur eine gerade Anzahl von Zeichen.</td></tr>
<tr><td>-sep</td><td>Beim Datentyp text ist die Angabe der folgenden Trennzeichen nicht erlaubt: ; = ( ) < > ␣ (also Strichpunkt, Gleichheitszeichen, runde Klammer auf und zu, Größerzeichen, Kleinerzeichen und Leerzeichen)</td></tr>
<tr><td>-temp-file</td><td>Die Angabe einer temporären Datei ist nicht erlaubt (siehe #datei bzw. @datei bei filename).</td></tr>
</table>

Zusatz	Bedeutung
without (Forts.)	
-user	Die Angabe einer Benutzerkennung ist nicht erlaubt.
-vers	Die Angabe der Version (siehe "datei(nr)") ist bei Banddateien nicht erlaubt.
-wild	Die Datentypen posix-filename bzw. posix-pathname dürfen keine Musterzeichen enthalten.
mandatory	Bestimmte Angaben sind für einen Datentyp zwingend erforderlich.
-corr	Eingabeformat: [[C]'][V][m]m.naso['] Angaben zum Datentyp product-version müssen den Korrekturstand (und damit auch den Freigabestand) enthalten.
-man	Eingabeformat: [[C]'][V][m]m.na[so]['] Angaben zum Datentyp product-version müssen den Freigabestand enthalten. Die Angabe des Korrekturstands ist optional möglich, wenn dies nicht durch den Zusatz without-corr untersagt wird.
-quotes	Angaben zu den Datentypen posix-filename bzw. posix-pathname müssen in Hochkommata eingeschlossen werden.

Operatoren

Operator	Operation	Operatortyp	Operand (Typ)	Ergebniswert
+	Addition	arithmetischer Operator	Integer	Integer
-	Subtraktion	arithmetischer Operator	Integer	Integer
*	Multiplikation	arithmetischer Operator	Integer	Integer
/	Division	arithmetischer Operator	Integer	Integer
MOD	Modulo	arithmetischer Operator	Integer	Integer
LT <	kleiner als	Vergleichs-Operator	Integer, String	Boolesche Konstante
LE <=	kleiner oder gleich	Vergleichs-Operator	Integer, String	Boolesche Konstante
EQ = ==	gleich	Vergleichs-Operator	Integer, String	Boolesche Konstante
NE <>	nicht gleich	Vergleichs-Operator	Integer, String	Boolesche Konstante
GE >=	größer oder gleich	Vergleichs-Operator	Integer, String	Boolesche Konstante
GT >	größer als	Vergleichs-Operator	Integer, String	Boolesche Konstante
NOT	Negation	logischer Operator	logischer-Ausdruck	Boolesche Konstante
OR	Oder	logischer Operator	logischer Ausdruck	Boolesche Konstante
AND	Und	logischer Operator	logischer Ausdruck	Boolesche Konstante
XOR	Exklusives Oder	logischer Operator	logischer Ausdruck	Boolesche Konstante
//	String-Verknüpfung	Verkettungs-Operator	String-Ausdruck	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.