Deutsch

Fujitsu Software

DSSM V21.0/SSCM V21.0
Subsystem Management in BS2000

Benutzerhandbuch

Stand der Beschreibung:
DSSM V21.0/SSCM V21.0

November 2025

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 bs2000.info@fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

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: 2015 erfüllt.

Copyright und Handelsmarken

Inhaltsverzeichnis

DSSM V21.0/SSCM V21.0

1 Einführung

Durch die Zerlegung des BS2000-Systems in funktionelle Einheiten (Subsysteme) wird die Komplexität des BS2000-Ablaufs reduziert. SSCM und DSSM sind die beiden Softwareprodukte, mit denen alle Subsysteme deklariert und verwaltet werden.

Mit SSCM (Static Subsystem Catalog Management) wird der statische Subsystemkatalog (SSMCAT) verwaltet. Im Subsystemkatalog werden die erforderlichen Deklarationen aller zu einer Konfiguration gehörenden Subsysteme abgelegt. Der SSMCAT bildet die Datengrundlage für DSSM.

DSSM (Dynamic Subsystem Management) ist die zentrale Instanz im BS2000-System zur dynamischen Verwaltung von Subsystemen.

1.1 Zielgruppe und Konzept des Handbuchs

Diese Handbuch wendet sich an die Systembetreuung des BS2000-Systems.

Das vorliegende Kapitel „**Einführung**“ enthält eine allgemeine Übersicht über das Handbuch und eine Zusammenstellung der Änderungen gegenüber dem Vorgängerhandbuch.

Das Handbuch ist in folgende Hauptkapitel gegliedert:

Im Kapitel „**Das Subsystem-Konzept in BS2000 OS DX**“ werden Begriffe erklärt und Zusammenhänge dargestellt, die für das Subsystem-Konzept eine wichtige Rolle spielen. Nach der Darstellung der Versionsabhängigkeiten zwischen BS2000, DSSM und SSCM folgt eine Übersicht über wichtige DSSM-fähige Produkte im BS2000-Grundausbau sowie ausgewählte entbündelte DSSM-fähige Produkte.

Im Kapitel „**DSSM**“ wird die Dynamische Subsystemverwaltung (DSSM) beschrieben: ihre Aufgaben und Funktionen, Verwaltungsstrategien und Fehlerbehandlung sowie die Inbetriebnahme des DSSM durch den Parameterservice zum Startup-Zeitpunkt und die Abrechnungssätze von DSSM.
Ein Beispiel schließt das Kapitel ab.

Das Kapitel „**SSCM**“ beschreibt die Subsystemkatalog-Verwaltung (SSCM). Nach einer kurzen Einführung werden die Anweisungen von SSCM ausführlich beschrieben.
Es folgen Hinweise zur Installation von SSCM und Beispiele.

Am Ende des Handbuchs finden Sie ein "**Literaturverzeichnis**".

Literaturhinweise werden im Text in Kurztiteln angegeben. Der vollständige Titel jeder Publikation, auf die verwiesen wird, ist im Literaturverzeichnis aufgeführt.

1.2 Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com.

1.3 Darstellungsmittel

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

> ¡ Für Hinweise auf besonders wichtige Informationen.

- Begriffe im Fließtext, die besonders hervorgehoben werden sollen, sind **halbfett** dargestellt.
- In den Beispielen sind die Benutzereingaben in **`halbfetter Schreibmaschinenschrift`** und Systemausgaben in `normaler Schreibmaschinenschrift` wiedergegeben.
- Literaturhinweise werden im Text in Kurztiteln angegeben. Der vollständige Titel jeder Druckschrift ist im Literaturverzeichnis aufgeführt.

2 Das Subsystem-Konzept in BS2000 OS DX

In diesem Kapitel werden wichtige Begriffe erläutert und anschließend wird informiert über

- die Verwaltung von Subsystemen mit DSSM und SSCM (siehe "Verwaltung von Subsystemen mit DSSM und SSCM")
- die wichtigen DSSM-fähigen Produkte im BS2000-Grundausbau (siehe "Übersicht über wichtige DSSM-fähige Produkte im BS2000-Grundausbau")
- ausgewählte entbündelte, DSSM-fähige Produkte (siehe "Übersicht über ausgewählte entbündelte, DSSM-fähige Produkte").

2.1 Erläuterung wichtiger Begriffe

Im BS2000 lassen sich neben globalen, systemweit gültigen Subsystemen auch lokale Subsysteme zu einer lokalen Subsystem-Konfiguration zusammenstellen und in einem lokalen Subsystemkatalog tasklokal verwalten. Wird im Folgenden von Subsystemen, Konfiguration und Katalog gesprochen, so sind immer die globalen, systemweit gültigen Subsysteme, deren Konfiguration und Katalog gemeint. Lokale Subsysteme, deren Konfiguration und Katalog werden immer mit dem Wort „lokal“ gekennzeichnet.

Subsystem

Ein Subsystem im Sinne der Dynamischen Subsystemverwaltung (DSSM) ist eine automatisch und unabhängig lad-, start- und beendbare Funktionsausführungseinheit, wobei Abhängigkeiten zu anderen Subsystemen beachtet werden müssen.
Es kann aus mehreren Subsystemkomponenten bestehen.

Name und Versionsnummer identifizieren ein Subsystem eindeutig gegenüber DSSM. Die Festlegung der Subsystemkomponenten und der Eigenschaften des Subsystems definieren ein Subsystem für DSSM.

Die Eigenschaften eines Subsystems (Attribute) liefern Informationen über:

- Identifikation (Name und Version)
- Zugriffsrechte
- Lade- und Bearbeitungsmechanismen
- Umgebung und Abhängigkeiten
- benötigte Adressräume
- Aufrufpräferenzen
- existierende Auftragseingänge

Die Beziehungen zwischen verschiedenen Subsystemen werden geprägt durch:

- Adressraum-Trennungen
- Bindebeziehungen
- funktionale Abhängigkeiten
- Holdertask-Sharing

Subsystemkomponenten

Unter Subsystemkomponenten sind das Subsystem-Modul (Programmodul) sowie die Nebenkomponenten zu verstehen.
Das Subsystem-Modul ist eine Komponente, die zur Installation des Subsystems unbedingt benötigt wird. Ein Subsystem kann aus verschiedenen Modulen aufgebaut sein.Nebenkomponenten für Subsysteme, die bei der Installation zu vereinbaren sind und beim Aktivieren benötigt werden, sind die Rep-Datei, die NOREF-Datei, die Meldungsdatei, die Syntaxdatei und die Informationsdatei.

Subsystem-Auftragseingang

Der Subsystem-Auftragseingang (call entry) ist die sichtbare Einsprungstelle, an der das Subsystem für Tasks aufrufbar ist.
Ein Auftragseingang wird bestimmt durch seinen Namen, den Typ, den Geltungsbereich und die Lebensdauer. Ein Subsystem kann mehrere Auftragseingänge mit unterschiedlichen Attributen haben. Sie werden von DSSM verwaltet.

Subsystem-Konfiguration

Eine Subsystem-Konfiguration ist die Menge aller Subsysteme, die in einem Systemlauf verfügbar sein sollen. Da immer nur eine Subsystem-Konfiguration im System bedient werden kann, muss sie alle Subsysteme enthalten, die gleichzeitig im System verfügbar sein sollen.

Definiert wird eine Subsystem-Konfiguration durch die Subsysteme, die zu dieser Konfiguration gehören, sowie deren Komponenten und Eigenschaften.

Ein Subsystem (bzw. eine Subsystemversion) kann innerhalb einer gegebenen Subsystem-Konfiguration und in einer bestimmten Kombination verträglich sein, d.h. das Subsystem kann in diesem System mit bestimmten anderen Subsystemen laufen.
Ein Subsystem kann optional oder zwingend sein, d.h. die Subsystem-Konfiguration ist ohne dieses Subsystem ablauffähig oder nicht.

Im laufenden System unterscheidet man zwei Zustände in einer Subsystem-Konfiguration:

1. Lade-Zustand:
   dieser ist gegeben durch die tatsächlich **aktiven** und damit auch wirklich nutzbaren Subsysteme
2. Deklarations-Zustand:
   dieser ist gegeben durch die deklarierten und damit potenziell **aktivierbaren** Subsysteme bzw. Subsystem-Versionen im Subsystemkatalog

Aus einem Deklarations-Zustand können mehrere verschiedene Lade-Zustände abgeleitet werden.

Subsystemkatalog

Die statische Subsystemkatalog-Verwaltung (SSCM) generiert einen statischen Subsystemkatalog (SSMCAT), in dem die Subsystem-Konfiguration definiert ist.
Dieser Subsystemkatalog wird durch die dynamische Subsystem-Verwaltung (DSSM) zum Zeitpunkt der Systemeinleitung geladen. Mit dem Laden wird aus dem statischen ein dynamischer Subsystemkatalog.
Während des Systemlaufs kann der Subsystemkatalog durch die DSSM-Kommandos ADD-SUBSYSTEM, MODIFY-SUBSYSTEM-PARAMETER und REMOVE-SUBSYSTEM verwaltet werden.

Sowohl SSCM als auch DSSM benutzen die Datenstruktur des statischen Subsystemkatalogs. Der Subsystemkatalog kann bis zu 1000 Subsysteme enthalten mit insgesamt max. 16000 Auftragseingängen, 16000 funktionalen Abhängigkeiten untereinander und 200 Adressraum-Einschränkungen.

2.2 Verwaltung von Subsystemen mit DSSM und SSCM

Die dynamische Verwaltung von Subsystemen (DSSM) verfolgt das Ziel, die Komplexität des BS2000-Ablaufs zu reduzieren. Dabei wird das Betriebssystem in funktionelle Einheiten (Subsysteme) zerlegt.

Als Subsysteme werden von DSSM unterstützt:

- Software-Produkte (soweit ausdrücklich als DSSM-fähig ausgewiesen, siehe auch die Tabellen 1 (Übersicht über wichtige DSSM-fähige Produkte im BS2000-Grundausbau) und 2 (Übersicht über ausgewählte entbündelte, DSSM-fähige Produkte))
- System-Exit-Routinen
- Share-Produkte

Darüber hinaus kann die Systembetreuung eigene TU-Programme und System-Exit-Routinen DSSM-fähig machen und als Subsysteme aktivieren.

DSSM unterscheidet in seinen Funktionen privilegiert ablaufende Subsysteme (TPR-Subsysteme), die Teil des BS2000-Systems sind, und nicht-privilegiert ablaufende Subsysteme (TU-Subsysteme), z.B. Benutzerprogramme. Ebenso unterscheidet DSSM danach, ob ein Subsystem im Systemadressraum oder im Benutzeradressraum geladen ist. TPR-Subsysteme sind immer im Systemadressraum geladen, TU-Subsysteme können entweder im System- oder im Benutzeradressraum geladen sein.

Das Subsystem SSCM ermöglicht eine flexible und anwenderfreundliche Verwaltung des statischen Subsystemkatalogs (SSMCAT).

Das SSD-Object

Das SSD-Object ist eine ISAM-Datei mit der Schlüssellänge von 11 Byte. Es enthält die Definition(en) eines oder mehrerer Subsysteme (jedoch nicht die Definitionen verschiedener Versionen ein und desselben Subsystems). Jede dieser Definitionen enthält Eigenschaften, Auftragseingänge, Referenzen und Abhängigkeiten des betreffenden Subsystems sowie Informationen über disjunkte Subsysteme und zur Holdertask.

Das SSD-Object selbst kann nicht geladen werden. Zur Aktivierung muss es mit der SSCM-Anweisung ADD-CATALOG-ENTRY in einen statischen Subsystemkatalog eingebunden werden.

2.3 Übersicht über wichtige DSSM-fähige Produkte im BS2000-Grundausbau

Sp-Kl: Speicherklassen (3-6), in die das Produkt geladen werden kann

TU/TPR: Funktionszustand des Subsystems (TU: nicht-privilegiert / TPR: privilegiert)

Produkt	Funktion	Sp-Kl	TU/TPR
ACS	Ermöglicht den Dateizugriff über Aliasnamen	3,4	TPR
AIDSYS	Systemabhängiger Teil der Test- und Diagnosehilfe AID (zu AID siehe Tabelle 2)	3,4	TPR
BINDER	Zusammenfügen eines übersetzten Quellprogramms mit anderen Objektmodulen und LLMs zu einer ladefähigen Einheit	4,6	TU
CALENDAR	Erstellung von benutzerspezifischen Kalendern	3,4	TPR
CCOPY			
DIV	Objektorientierte Zugriffsmethode, besonders zur Verarbeitung unstrukturierter Daten	3,4	TPR
FASTPAM	Blockorientierte Zugriffsmethode für NK4-Plattendateien	3,4	TU/TPR
GET-TIME	Bereitstellen von Datum und standardisierter Weltzeit und lokaler Landeszeit	4	TU
IMON	Monitor zur Softwareinstallation		
IMON	(IMON-BAS) Installation und Registration von Software	4	TPR
IMON-GPN	Unterstützung der Zuordnung von logischen Namen und Pfadnamen von Dateien	4	TPR
INIT	Initialisieren von Magnetbändern, MBKs und Disketten	4,6	TU/TPR
MIP	Verwalten und Ausgeben von Systemmeldungen	4	TPR
NKS	Überwachen der Betriebsmittelreservierung	4	TPR
NKV	Überwachen der montierten Datenträger	4	TPR
NKISAM	Satzorientierte Zugriffsmethode für Dateien mit dem Blockformat "DATA" (ohne gesonderten PAM-Schlüssel)	4	TPR
PAMCONV	Konvertieren von Dateiformaten	4,6	TU
POSIX-BC	GA-Komponente von POSIX	3,4	TPR
POSPRRTS	Runtime-Funktionen für C (in TPR)	4	TPR
REWAS			
SDF	System Dialog Facility; Sprachoberfläche des BS2000	4	TPR
SDF-CONV	Konvertieren von Prozedurformaten	6	TU

SDF-P-BASYS	GA-Komponente des Produkts SDF-P (zu SDF-P siehe Tabelle 2)	4	TPR
SECOS-KRB			
SHOW-FILE	Ausgabe von Dateiinhalten auf den Bildschirm	3,4	TPR
SIR	Installieren von Software	4	TPR
SPOOL	Ein/Ausgabe-Steuerung auf bestimmte Gerätefamilien		
PRMPRES	Erstellen und Verwalten von Druckressourcen im BS2000-SPOOL (Komponente von PRM)	4	TU
PRMMAN	Erstellen und Verwalten von Druckressourcen im BS2000-SPOOL (Komponente von PRM)	4	TPR
SPOOL	Organisation von SPOOLIN/SPOOLOUT und Verwalten von Druckaufträgen	4	TPR
SRPMNUC	System Resources and Privilege Management (GA-Komponente von SRPM; zu SECOS siehe Tabelle 2)	4	TPR
SSCM	Generieren von Subsystemkatalogen	4,6	TU, siehe hier
SYSFILE	Unterstützen der Systemdateien SYSLST und SYSOUT	4	TPR
TANGRAM	Zuordnung affiner Taskgruppen zu Prozessoren unter Berücksichtigung ihres Leistungsbedarfs		
TANGRAM	Regelungsfunktion	3,4	TPR
TANGBAS	Verwaltung der Taskgruppen	3,4	TPR
VOLIN	Formatieren und Initialisieren von Festplatten	4,5	TPR

Tabelle 1: DSSM-fähige Produkte im Grundausbau (GA)

2.4 Übersicht über ausgewählte entbündelte, DSSM-fähige Produkte

Sp-Kl: Speicherklassen (3-6), in die das Produkt geladen werden kann

TU/TPR: Funktionszustand des Subsystems (TU: nicht-privilegiert / TPR: privilegiert)

Produkt	Funktion	Sp-Kl	TU /TPR
AID	Test- und Diagnosehilfe		
AID	Symbolisches und nicht-symbolisches Testen	4	TPR
LLMAID	Information über Bindelademodule	4	TPR
ARCHIVE	Sichern, Rekonstruieren und Transfer von Daten in Dateien und Jobvariablen	4	TPR
CRTE	Runtime-Funktionen für C und COBOL	4,6	TU
DAB	Caching im BS2000 zur Vermeidung von Performance-Engpässen (Software-Caching)	3,4	TPR
Dprint	(Distributed Print Services) Drucken in Computernetzen		
DPRINTCL	Client-Part für Dprint: Erzeugen von Druckaufträgen	4	TPR
DPRINTCM	Basismechanismen: Realisierung allgemeiner Dienste	4	TPR
DPRINTSV	Server-Part für Dprint: Verwalten von Druckaufträgen	4	TPR
DRV	Aufzeichnungsverfahren, mit dem Daten auf zwei Platten doppelt geführt werden können	4	TPR
EDT	Editor für SAM-, ISAM- und POSIX-Dateien sowie für Elemente aus Programmbibliotheken	4,6	TU
FDDRL	Physikalische Datensicherung von Platten und Pubsets	4,5	TPR
HIPLEX MSCF	Einrichten eines Rechnerverbundes auf der Basis des Datenkommunikationsnetzes BCAM	3,4	TPR
HSMS	Sichern, Archivieren und Rekonstruieren von Daten sowie Unterstützen der Datenverwaltung auf externen Speichern	4	TPR
JV	Steuern von Aufträgen und Programmen mit Jobvariablen	4	TPR
LMS	Verwalten von Bibliotheken	4,6	TU
MAREN	Verwalten von Datenträgerbeständen im BS2000-RZ	3,4	TPR
PCS	Unterstützen des Systembetreuers bei der optimalen Einstellung und dem optimalen Betrieb der BS2000-Anlage	3,4	TPR
PROP-XT	RZ-Automatisierung und benutzerspezifische Problemlösung im RZ	4	TPR

RSO	Steuern der Ausgabe von Fern-SPOOL-Aufträgen auf dezentrale (RSO-) Drucker (nur mit SPOOL ablauffähig)	4	TPR
SDF-A	Verwalten und Anpassen der SDF-Benutzeroberfläche	6	TU
SDF-P	Prozedur- und Variablenkonzept, Kommandos zur Prozedursteuerung, blockorientierte Fehlerbehandlung	4	TPR
SDF-P-BIF	Builtin-Funktionen von SDF-P	4	TPR
SECOS	Gewährleisten der Zugriffssicherheit für das BS2000		
GUARDS	Verwalten von Objekten (Guards) und Auswerten der Zugriffsbedingungen (Komponente von GUARDS)	4	TPR
GUARDDEF	Verwalten des Standardschutzes sowie der Attribut- und Objektpfade	4	TPR
GUARDCOO	Verwalten des Miteigentümerschutzes	4	TPR
SATCP	Überwachen von Ereignissen und Alarmen (Komponente von SAT)	4	TPR
SRPMOPT	(Komponente von SRPM)	4	TPR
SM2	Messsystem zum Erfassen und Auswerten von Daten zur Leistung des BS2000 und zur Auslastung der Betriebsmittel	3,4	TPR

Tabelle 2: DSSM-fähige entbündelte Produkte (Auswahl)

3 DSSM

Die dynamische Verwaltung von Subsystemen (DSSM) verfolgt das Ziel, die Komplexität des BS2000-Ablaufs zu reduzieren. Dabei wird das Betriebssystem in funktionelle Einheiten (Subsysteme) zerlegt.

Subsysteme besitzen folgende Eigenschaften:

- Jedes Subsystem bildet eine in sich geschlossene Einheit.
- Sie können während des BS2000-Systemlaufs aktiviert, angehalten, fortgesetzt und deaktiviert werden.
- Ist für ein Subsystem eine neue Version erstellt worden, kann die alte gegen die neue Version ausgetauscht werden oder beide Versionen können parallel betrieben werden.

DSSM verwaltet als zentrale Instanz die Subsystemkonfiguration im BS2000.
Es steuert das Laden, Initialisieren, Anhalten, Fortsetzen und Beenden von Subsystemen im Systemlauf. DSSM kann das aktuelle System dynamisch (online) verändern, indem es Subsysteme in den Subsystemkatalog einbinden oder aus ihm herausnehmen kann, ohne dass das gesamte System angehalten und wieder neu gestartet werden muss.

DSSM ist für BS2000 zwingend, da wichtige Bestandteile des Grundausbaus (siehe die Produktübersicht auf "Übersicht über wichtige DSSM-fähige Produkte im BS2000-Grundausbau") nur mit DSSM betrieben werden können.

Die DSSM-Privilegierung für Kommandos

Im Rahmen des BS2000-Privilegienkonzeptes existiert ein systemglobales Privileg für die Ausführung der DSSM-Kommandos. Das Privileg zur Subsystem-Verwaltung wird in Kommandos und Meldungen mit SUBSYSTEM-MANAGEMENT bezeichnet und ist nach First-Startup der Benutzerkennung TSOS zugeteilt (Standard).
Das Privileg kann bei Einsatz von SECOS an jede beliebige Benutzerkennung vergeben werden, die dann die alleinige Berechtigung besitzt, diese Kommandos ausführen zu lassen (siehe auch Handbuch „SECOS“).

Die nachfolgende Tabelle enthält eine Übersicht mit allen DSSM-Kommandos und den zur Kommandoausführung erforderlichen Privilegien.

Kommando	**Bedeutung**	**nötiges Privileg**			
		SM	**O**	**STD**	**HM**
ADD-SUBSYSTEM	Dynamischen Subsystemkatalog erweitern	X			
HOLD-SUBSYSTEM	Subsystem in den Wartezustand versetzen	X	X		
MODIFY-SUBSYSTEM-PARAMETER	Subsystem-Parameter verändern	X			
RELEASE-SUBSYSTEM-SPACE	Reservierten Adressraum von Subsystemen freigeben	X		X	X
REMOVE-SUBSYSTEM	Inaktives Subsystem aus dynamischem Katalog entfernen	X			

RESUME-SUBSYSTEM	Wartezustand für ein Subsystems aufheben	X	X		
SAVE-SUBSYSTEM-CATALOG	Änderungen des dynamischen Subsystemkatalogs sichern	X			
SET-DSSM-OPTIONS	Protokollierungsfunktion von DSSM aus- oder einschalten	X	X		
SHOW-DSSM-INFORMATION	Informationen über DSSM anzeigen	X			
SHOW-RESTART-OPTIONS	Informationen über den automatischen Restart ausgeben		X		
SHOW-SUBSYSTEM-ATTRIBUTES	Informationen über Subsystem-Attribute anzeigen	X		X	
SHOW-SUBSYSTEM-INFO	Informationen über aktuelle Subsystem-Konfiguration anfordern	X			
SHOW-SUBSYSTEM-STATUS	Informationen über den Zustand von Subsystemen anfordern	X			
subsys=*NON-PRIV-CLASS-5				X	
subsys=*ALL / <subsys-name>			X	X	
START-SUBSYSTEM	Subsystem aktivieren	X	X		
STOP-SUBSYSTEM	Subsystem deaktivieren	X	X		
UNLOCK-SUBSYSTEM	Subsystem im Zustand LOCKED in den Zustand NOT-CREATED überführen	X			

Tabelle 3: DSSM-Kommandos und ihre Privilegierung

SM: Privileg SUBSYSTEM-MANAGEMENT zur Verwaltung von Subsystemen
O: Privileg OPERATING zur Eingabe von Bedienplätzen
STD: Privileg STD-PROCESSING zur Ausführen von Benutzerkommandos
HM: Privileg HARDWARE-MAINTENANCE zum Aufruf von Hardware-Testprogrammen

3.1 Aufgaben und Funktionen von DSSM

Das Softwareprodukt DSSM führt folgende Aktionen durch:

- DSSM baut die gesamte Subsystem-Konfiguration bis zur Funktionsfähigkeit auf.
- DSSM aktiviert und deaktiviert Subsysteme, um die Subsystem-Konfiguration zu erweitern bzw. an die aktuellen Anforderungen anzupassen.
- DSSM überprüft die Verträglichkeit der einzelnen Subsystemversionen beim Aktivieren und Deaktivieren von Subsystemen.
- DSSM stellt die Bezüge zwischen Subsystemen her und überwacht diese beim Aktivieren bzw. hebt vorhandene Beziehungen eines Subsystems zu anderen Subsystemen beim Deaktivieren auf.
- DSSM unterstützt den Anschluss der Tasks und Programme an Subsysteme.
- DSSM unterstützt die Nutzung gemeinsamen Adressraums in Memory Pools.

Alle in den folgenden Punkten als synchron bezeichnete Aktivitäten laufen in der Task des Auftraggebers ab, alle asynchronen Aktivitäten in der Holdertask.

3.1.1 Subsystem-Deklaration (SSC)

Zur Verwaltung der Subsysteme benötigt DSSM Informationen in Form einer so genannten Deklaration. Die Deklaration bestimmt, welche Haupt- und Nebenkomponenten zu einem Subsystem gehören, wie es aktiviert werden kann und wie die Anschlüsse an das Subsystem hergestellt werden müssen. Die Erstellung der Deklaration erfordert also die Kenntnis des Subsystems. Deshalb werden für alle DSSM-fähigen Subsysteme die dafür notwendigen Deklarationen in einer Deklarationsdatei mit ausgeliefert.

Abhängig von der Deklaration aktiviert DSSM das Subsystem und stellt Anschlüsse und Bezüge zu anderen Subsystemen und zu DSSM her. Anschlüsse an ein Subsystem können von DSSM überwacht werden. Dadurch ist ein Deaktivieren des Subsystems unter Steuerung von DSSM möglich. Werden die Anschlüsse nicht überwacht, erfolgt das Deaktivieren in eigener Verantwortung des Subsystems.

Die Deklarationen werden nicht jedem DSSM-Aufruf als Operanden mitgegeben, da dies zu umständlich und umfangreich wäre. Die Deklarationen werden mit SSCM festgelegt und im Subsystemkatalog abgespeichert, weil:

- diese Deklarationen stabil sind und sich selten ändern
- diese Deklarationen umfangreich und komplex sind
- auf diese Deklarationen häufig von DSSM zugegriffen wird
- nicht nur subsystem-interne, sondern auch subsystem-übergreifende Informationen existieren

Die Deklarationen aller zu einer Konfiguration gehörenden Subsysteme („Deklarations-Zustand“) bilden den Subsystemkatalog SSMCAT und werden in einer PAM-Datei abgelegt. Diese Datei wird beim Startup eingelesen und bleibt bis zur Systembeendigung im Speicher. Der Katalog kann im laufenden System mit dem Kommando ADD-SUBSYSTEM erweitert werden.

Mit dem Befehl LOAD-LOCAL-SUBSYSTEM-CATALOG kann der Katalog gelesen und dynamisch in den Benutzeradressraum der aufrufenden Task geladen werden.

3.1.2 Aktivieren bzw. Wiederanlauf

Die **Aktivierung** eines Subsystems wird gesteuert durch die Attribute, die im Subsystemkatalog mit der SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES definiert wurden.

Die Aktivierung erfolgt durch eine der folgenden Möglichkeiten:

- automatisch während des ersten DSSM-Aufrufs der Startup-Routine, wenn CREATION-TIME=*BEFORE-DSSM-LOAD oder *AT-DSSM-LOAD definiert wurde; die Aktivierung muss dabei erfolgreich abgeschlossen werden (*BEFORE-DSSM-LOAD bedeutet dabei, dass das Subsystem bereits geladen ist, bevor DSSM erstmals durch die Startup-Routine aufgerufen wurde. Nach dem Aufruf von DSSM wird dieses Subsystem wie jedes andere verwaltet. Versionsaustausch oder Versionskoexistenz mit einer anderen Version, die mit CREATION-TIME=*AT-DSSM-LOAD definiert wurde, ist erlaubt. Es liegt in der Verantwortung der Systembetreuung dafür zu sorgen, dass zu jeder Zeit eine Version des Subsystems verfügbar ist.)
- automatisch während des zweiten DSSM-Aufrufs der Startup-Routine, wenn CREATION-TIME=*MANDATORY-AT-STARTUP definiert wurde; die Aktivierung muss dabei erfolgreich abgeschlossen werden
- automatisch während des zweiten DSSM-Aufrufs der Startup-Routine, wenn CREATION-TIME=*BEFORE-SYSTEM-READY definiert wurde (synchron)
- automatisch nach dem zweiten Rücksprung zur Startup-Routine, wenn CREATION-TIME=*AFTER-SYSTEM-READY definiert wurde (asynchron)
- explizit durch das Kommando START-SUBSYSTEM
- explizit beim Aufruf der $ESMCRE-Schnittstelle
- implizit nach dem ersten SVC-Aufruf für ein Subsystem, das mit CREATION-TIME=*AT-SUBSYSTEM-CALL(ON-ACTION=*STD oder *ANY) definiert wurde bzw. implizit nach dem ersten ISL-Aufruf für ein Subsystem, das mit CREATION-TIME= *AT-SUBSYSTEM-CALL(ON-ACTION=*ISL-CALL oder *ANY) definiert wurde

Zum Startzeitpunkt für Subsysteme siehe auch die SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES.

Die Programm- oder Bindemodulbibliothek (OML), die REP- und die „NOREF“-Datei von DSSM müssen ebenso wie die anderen für die DSSM-Initialisierung benötigten Dateien (z.B. der SSMCAT) und die Dateien, die für die Aktivierung von Startup-gebundenen Subsystemen benötigt werden, unter der Kennung TSOS auf dem Home-Pubset eingerichtet und während des Startup verfügbar sein.

Startup-gebunden sind solche Subsysteme, deren Aktivierungszeitpunkt mit den folgenden Werten angegeben wurden:

- *BEFORE-SYSTEM-READY
- *AFTER-SYSTEM-READY
- *BEFORE-DSSM-LOAD
- *AT-DSSM-LOAD
- *MANDATORY-AT-STARTUP

Voraussetzung für das Aktivieren ist in jedem Fall, dass das Subsystem bekannt, d.h. im globalen oder lokalen Subsystemkatalog deklariert ist.

Die Aktivierung läuft in folgenden Stufen ab:

1. Prüfen des Auftrags, insbesondere Abhängigkeiten zu anderen Subsystemen (synchron)

   Hat ein Subsystem Komponenten, die mit dem Attribut *INSTALLED deklariert und mit IMON installiert wurden, ruft DSSM während der Prüfphase IMON-GPN auf, um die Pfadnamen der Dateien zu erhalten.

   Abhängig von der Verfügbarkeit von IMON-GPN und dem Status der installierten Liefergruppe (siehe Operand INSTALLATION-UNIT) verhält sich DSSM wie folgt:

   - Die INSTALLATION-UNIT existiert und ein Dateiname kann mit dem angegebenen logischen Namen assoziiert werden:
     DSSM wird diesen Dateinamen solange benutzen, wie das Subsystem aktiv ist (bis STOP-SUBSYSTEM) oder bis durch das Kommando MODIFY-SUBSYSTEM-PARAMETER ein anderer Dateiname festgelegt wurde.
   - In folgenden Situationen greift DSSM auf den im Katalog definierten DEFAULT-NAME zu, wenn vorhanden:
     - IMON-GPN ist nicht verfügbar
     - die INSTALLATION-UNIT existiert nicht in den IMON-GPN-Daten
     - die INSTALLATION-UNIT existiert in den IMON-GPN-Daten, jedoch ohne Verbindung zu einem logischen Namen (LOGICAL-ID)

   Die Meldung `ESM0665` informiert über die Vorgehensweise:

   ```
   ESM0665  'DEFAULT-NAME' FUER DATEIEN DES SUBSYSTEMS '(&00)' VERWENDET
   ```

   - Die INSTALLATION-UNIT existiert in den IMON-GPN-Daten mit einigen zugehörigen Pfadnamen, aber der angefragte logische Name existiert nicht. DSSM nimmt nun an, dass die Datei nicht existiert. Zwei Fälle sind zu unterscheiden:
     - Subsysteme, die mit den Attributen *AT-DSSM-LOAD oder *MANDATORY-AT-STARTUP definiert wurden:
       Während dem Startup wird eine Frage an der Bedienstation ausgegeben, die beantwortet werden muss. Der Operator kann nun entweder für die erforderliche Datei (Informationsdatei, Modulbibliothek oder Rep-Datei mit dem Attribut REP-FILE-MANDATORY=*YES) einen neuen, gültigen Namen angeben oder die Aktivierung des Subsystems stoppen.
     - Subsysteme, die mit den Attributen *AT-CREATION-REQUEST, *AFTER-SYSTEM-READY oder *BEFORE-SYSTEM-READY definiert wurden:
       Fehlt eine der Dateien (Modulbibliothek, Informationsdatei oder Rep-Datei mit dem Attribut REP-FILE-MANDATORY=*YES), wird das Subsystem nicht aktiviert, was durch eine Meldung angezeigt wird. Fehlt die Meldungsdatei oder die Rep-Datei (mit Attribut REP-FILE-MANDATORY=*NO), wird das Subsystem aktiviert; es wird trotzdem eine Warnung ausgegeben.

2. Laden in eine Holdertask (asynchron)
   Das Laden des Subsystemcodes wird durch die Holdertask veranlasst.
   Zusätzlich stellt sie ihren lokalen Adressraum (Benutzeradressraum) für die Subsysteme zur Verfügung, die mit MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED definiert wurden. Das Laden von Subsystemen mit MEMORY-CLASS=*SYSTEM-GLOBAL wird zwar ebenfalls von der Holdertask veranlasst, sie werden aber nicht in den Benutzeradressraum der Holdertask, sondern in den gemeinsam benutzbaren Systemadressraum geladen.

   Bei der Aktivierung von TU-Subsystemen mit MEMORY-CLASS=*BY-SLICE wird der mehrbenutzbare Programmbereich in den gemeinsam benutzbaren
   Systemadressraum geladen, der nicht-mehrbenutzbare Programm- und/oder Datenbereich in den Benutzeradressraum der Holdertask.

Stellt eine Task den Anschluss zu einem solchen Subsystem her, wird nur noch der bereits im Benutzeradressraum der Holdertask geladene Datenbereich in die privaten Benutzeradressräume der angeschlossenen Tasks an dieselbe Adresse kopiert. Dadurch sind adressbezogene Referenzen zwischen Programm- und Datenbereich immer möglich. Die Performance wird durch diese Art der Adressraumaufteilung wesentlich erhöht, weil beim Anschluss einer Task an das Subsystem kein Zugriff zur Programm- oder Bindemodulbibliothek erfolgt und Externverweise nicht aufgelöst werden müssen.

Wenn ein Subsystem mit MEMORY-CLASS=*BY-SLICE definiert wurde und erstmals gestartet wird, informiert DSSM das Subsystem BLSSERV darüber, dass mit dem Makro VSVI1 auf die Kopie des Datenbereichs im privaten Benutzeradressraum zugegriffen werden kann.
Das Makro VSVI1 informiert den Anwender über Einträge in den Tabellen des DBL. Einzelheiten zum Makro siehe Handbuch „BLSSERV“.

Die Holdertask kann auch als Arbeitstask dienen.

3. nur für TPR-Subsysteme: Starten der Subsystemaktivierung in der Holdertask, falls eine INIT-Routine definiert ist (asynchron)
4. Nach abgeschlossener Aktivierung Öffnen des Subsystems für Anschlüsse von berechtigten Benutzern (asynchron)

Der **Wiederanlauf** eines Subsystems (RESUME-SUBSYSTEM) nach einem vorhergehenden HOLD-SUBSYSTEM besteht aus den Stufen 1, 3 und 4.
Bei Stufe 1 entfällt das Aufrufen von IMON-GPN.

Wenn die Aktivierung bzw. der Wiederanlauf eines Subsystems explizit durch das Kommando START- bzw. RESUME-SUBSYSTEM erfolgte, kann an Stelle des asynchronen der synchrone Verarbeitungsmodus gewählt werden.

3.1.3 Anschluss herstellen und wieder aufheben

Der **Anschluss** an den Auftragseingang eines Subsystems erfolgt durch eine der folgenden Möglichkeiten:

1. implizit und global durch Bindebeziehung; dies ist nur zu bereits geladenen Subsystemen oder zwischen gleichzeitig geladenen Subsystemen möglich
2. implizit und taskspezifisch
   - durch eine subsystemspezifische SVC-, ISL- bISL- bzw. System-Exit-Routine
   - bei nicht-privilegierten Subsystemen durch DBL-Schnittstellen (BIND-Makro, START-EXECUTABLE-PROGRAM, LOAD-EXECUTABLE-PROGRAM und START-<produktname>, Autolink)
3. explizit und taskspezifisch durch systeminterne Makros ($ESMCON und $ESMCCS)

Der Anschluss an ein Subsystem umfasst:

1. Erzeugen einer Subtask, wenn entsprechender systeminterner Aufruf ($ESMCCS)
2. Anschließen an den Memory Pool, falls erforderlich
3. Wenn ein Subsystem mit MEMORY-CLASS=*BY-SLICE definiert wurde, wird der private Bereich in den lokalen Adressraum (Benutzeradressraum) kopiert. Wird das Subsystem erstmals gestartet, informiert DSSM das Subsystem BLSSERV darüber, dass mit dem Makro VSVI1 auf die Kopie im privaten Benutzeradressraum zugegriffen werden kann.
4. Übergeben der Adresse an die Task oder Aufruf des Subsystemcodes
5. Setzen des subsystemspezifischen Anschlussmerkers
6. Erhöhen des taskspezifischen Anschlusszählers (Ausnahme: bei Definition des Subsystems mit dem Attribut CONNECTION-SCOPE=*FREE)

Es kann auch ein Anschluss für Auftragseingänge hergestellt werden, die das Subsystem selbst verwaltet.

Das **Aufheben** eines Anschlusses erfolgt durch eine der folgenden Möglichkeiten:

- implizit und taskspezifisch
  - durch Programm- bzw. Task-Beendigung
  - nach Rücksprung aus dem Subsystem-Auftragseingang, wenn dieser mit dem Attribut MODE=*SVC/*ISL und CONNECTION-SCOPE=*CALL definiert wurde
  - bei der Deaktivierung von Subsystemen mit CONNECTION-SCOPE=*OPTIMAL
- explizit und taskspezifisch durch einen systeminternen Makroaufruf bei Beendigung einer SVC-Routine ($ESMDCN)
- implizit und subsystemspezifisch bei Deaktivierung eines Subsystems, das mit dem Attribut CONNECTION-SCOPE=*FREE definiert wurde

Das Aufheben einer Beziehung enthält:

- Weghängen vom Memory Pool
- Wenn das Subsystem mit MEMORY-CLASS=*BY-SLICE definiert wurde, wird der Anteil aus dem Benutzeradressraum der angeschlossenen Task entladen.
  DSSM informiert das Subsystem BLSSERV beim Aufheben der letzten Verbindung darüber, dass nicht mehr auf diesen privaten Anteil zugegriffen werden kann.
- Zurücksetzen des taskspezifischen Anschlussmerkers

- Erniedrigen des subsystemspezifischen Anschlusszählers (Ausnahme: bei Definition des Subsystems mit dem Attribut CONNECTION-SCOPE=*FREE)
- „Function detach“ bei explizitem Aufruf

Die beim Anschluss an die Task gelieferte Adresse kann nicht mehr verwendet werden.

3.1.4 Subsystem deaktivieren oder anhalten

Voraussetzung für diese Funktionen ist generell, dass das Subsystem aktiv ist.

Das **Deaktivieren** eines Subsystems erfolgt durch eine der folgenden Möglichkeiten:

1. explizit bei Eingabe des Kommandos STOP-SUBSYSTEM (bei HOLD-SUBSYSTEM wird das Subsystem angehalten)
2. explizit bei Aufruf privilegierter Makros an der Programmschnittstelle ($ESMDEL und $ESMHLD)
3. implizit bei Eingabe des Kommandos START-SUBSYSTEM mit dem Operanden VERSION-PARALLELISM=*EXCHANGE-MODE
4. automatisch bei Shutdown für alle Subsysteme, bei deren Definition dies vereinbart wurde (STOP-AT-SHUTDOWN=*YES)

Das Deaktivieren eines Subsystems (STOP-SUBSYSTEM) läuft in folgenden Stufen ab:

1. Prüfen des Auftrags, insbesondere der Abhängigkeiten zu anderen Subsystemen (synchron)
2. Starten der CLOSE-CTRL-Routine, sofern eine solche definiert ist (asynchron)
3. Schließen des Subsystems für neue Benutzer (asynchron); so können keine Tasks mehr an das Subsystem angeschlossen werden;
   Ausnahme: Einsprungstellen mit CONNECTION-SCOPE=*FREE oder SVC/ISL-Aufrufe für Subsysteme mit Einsprungstellen mit CREATION-TIME=*AT-SUBSYSTEM-CALL. Auf Code, dessen Einsprungstellen mit CONNECTION-SCOPE=*OPTIMAL definiert wurde, kann von diesem Moment an nicht mehr zugegriffen werden.
4. Starten der Auftragsbeendigungs-Routine (STOPCOM-Routine), falls definiert (asynchron)
5. Warten, bis das Subsystem auftragslos ist, d.h. der subsystemspezifische Anschlusszähler gleich null ist und keine Task auf Code zugreift, dessen Einsprungstellen mit CONNECTION-SCOPE=*OPTIMAL definiert wurde (asynchron); Ausnahme: Wenn im DSSM-Kommando der Operand FORCED=*YES angegeben ist, wird die Deinitialisierung sofort gestartet.
6. Starten der Deinitialisierung, falls definiert (asynchron)
7. Entladen aus der Holdertask (asynchron)

Das **Anhalten** eines Subsystems (HOLD-SUBSYSTEM) besteht aus den Stufen 1 bis 6. Wenn die Deaktivierung bzw. das Anhalten eines Subsystems explizit durch ein Kommando (STOP-/HOLD-SUBSYSTEM) erfolgte, kann an Stelle des asynchronen der synchrone Verarbeitungsmodus gewählt werden.

3.1.5 Austausch von Subsystemversionen

Der Austausch von Versionen eines Subsystems ist auf folgende drei Arten möglich:

1. Die alte Version wird deaktiviert (STOP-SUBSYSTEM) und die neue Version des Subsystems aktiviert (START-SUBSYSTEM). Es kann u.U. zu sehr langen Ausfallzeiten des Subsystems kommen, weil die neue Version erst dann aktiviert wird, wenn die alte vollständig deaktiviert ist, d.h. alle Tasks die Verbindung abgebaut haben.
2. Die neue Version des Subsystems wird mit START-SUBSYSTEM ...,VERSION-PARALLELISM=*EXCHANGE-MODE aktiviert. Ab diesem Zeitpunkt werden keine neuen Verbindungen zur alten Version des Subsystems mehr hergestellt. Während die letzten Tasks die Verbindung zur alten Version des Subsystems abbauen, wird die neue Version initialisiert. Dieses Verfahren verkürzt den Zeitraum der Nichtverfügbarkeit des Subsystems erheblich.
   Es werden nacheinander folgende Routinen aufgerufen, sofern sie definiert sind:
   - STOPCOM-Routine der alten Version
   - INIT-Routine der neuen Version
   - DEINIT-Routine der alten Version

   Für manche Subsysteme ist es problematisch, dass die DEINIT-Routine der alten Version läuft, während die neue Version bereits aktiviert ist und arbeitet. Ein Subsystem, das mit VERSION-EXCHANGE=*FORBIDDEN definiert wurde, kann nicht als neue Version eingetauscht werden. Es kann jedoch (als alte Version) im Austausch gegen eine neue Version, die mit VERSION-EXCHANGE=*ALLOWED definiert ist, deaktiviert werden.

   Die alte Version ist so lange im Zustand IN-DELETE, bis keine Task mehr Anschluss hat. Ist die neue Version im Zustand CREATED, ist die Aktivierung des alten Subsystems mit RESET=*YES nur dann möglich, wenn für beide Versionen bei der Definition Koexistenz zugelassen wurde.

   Die Aktivierung der alten Version mit RESET=*YES ist dann erlaubt, wenn die neue Version im Zustand IN-DELETE ist, und die alte Version nicht mit VERSION-EXCHANGE=*FORBIDDEN definiert wurde.
3. Bei Verwendung der CLOSE-CTRL-Routine ist ein Versionsaustausch auch ohne Unterbrechung der Subsystemverfügbarkeit möglich.
   Es werden nacheinander folgende Routinen aufgerufen, sofern sie definiert sind:
   - CLOSE-CTRL-Routine der alten Version
   - INIT-Routine der neuen Version
   - STOPCOM-Routine der alten Version
   - DEINIT-Routine der alten Version

   Läuft die Initialisierungsroutine der neuen Version nicht fehlerfrei, wird die alte Version wieder automatisch aktiviert und ist im Zustand CREATED (das Ergebnis der CLOSE-CTRL-Routine ist reversibel). So wird eine Unterbrechung der Subsystemverfügbarkeit vermieden.

   Der Versionsaustausch ist erlaubt, wenn eine Version des Subsystems im Zustand CREATED ist und alle anderen im Katalog deklarierten Versionen des Subsystems im Zustand NOT-CREATED oder LOCKED sind. Ein Versionsaustausch wird nicht ausgeführt, wenn alle deklarierten Versionen des Subsystems im Zustand NOT-CREATED oder LOCKED sind. In diesem Fall wird die Version aktiviert, die im START-SUBSYSTEM-Kommando angegeben wurde.

Beispiel

Die Subsystemversionen SPOOL V04.2A und V04.3A sind mit VERSION-EXCHANGE=*ALLOWED definiert.

```
/SHOW-SUBSYSTEM-STATUS SPOOL,*ALL
SUBSYSTEM SPOOL    /V04.2.A IS NOT CREATED
SUBSYSTEM SPOOL    /V04.3.A IS NOT CREATED
/START-SUBSYSTEM SPOOL,04.2.A,VERSION-PARAL=*EXCHANGE-MODE,SYNCH=*YES
ESM0220  FUNCTION 'CREATE' FOR SUBSYSTEM 'SPOOL /V04.3.A' COMPLETELY
         PROCESSED
ESM0400  'CREATE' OR 'RESUME' SUBSYSTEM 'SPOOL /V04.3.A' WITH
         'SYNCHRONOUS=YES' AND 'RESET=NO'
ESM0220  FUNCTION 'CREATE' FOR SUBSYSTEM 'SPOOL /V04.3.A' COMPLETELY
         PROCESSED
```

Gibt es eine oder mehrere Versionen des Subsystems, die nicht im Zustand NOT-CREATED oder LOCKED sind (außer der auszutauschenden Version im Zustand CREATED), wird der Austausch abgewiesen, auch wenn alle Versionen Koexistenz erlauben.

Beispiel

Die Subsystemversionen UTM V06.3, V06.4 und V06.5 sind mit VERSION-COEXISTENCE=*ALLOWED und VERSION-EXCHANGE=*ALLOWED definiert.

```
/SHOW-SUBSYSTEM-STATUS UTM,*ALL
SUBSYSTEM UTM    /V06.5    IS NOT RESUMED
SUBSYSTEM UTM    /V06.5    IS NOT CREATED
SUBSYSTEM UTM    /V06.5    IS CREATED
/START-SUBSYSTEM UTM,06.4,VERSION-PARALLELISM=*EXCHANGE-MODE
ESM0206  SOME ACTIONS IN PROGRESS FOR SUBSYSTEM 'UTM/V06.3'.
         NO FURTHER ACTION ON ANOTHER VERSION POSSIBLE
ESM0224  REQUESTED FUNCTION 'CREATE' FOR SUBSYSTEM 'UTM/V06.4'
         REJECTED
```

Während des Austauschs der alten durch die neue Version des Subsystems wird auch die Syntaxdatei der neuen Version geladen. Das bedeutet, dass die Syntax der neuen Version auch Kommandos und Anweisungen der alten Version erkennen und ausführen muss. Es muss also darauf geachtet werden, dass die Syntax der neuen Version die der alten Version kompatibel unterstützt.

Es wird empfohlen, den Austausch von Versionen unter Verwendung der CLOSE-CTRL-Routine nur dann auszuführen, wenn die neu zu aktivierende Version auch die höhere Version von beiden ist.

3.1.6 Koexistenz von Versionen

Es gibt die Möglichkeit, temporär oder permanent zwei Versionen eines Subsystems aktiv zu halten, um allen Tasks, die an die alte Version angeschlossen sind, die Fortsetzung ihrer Arbeit zu gewährleisten. Der Koexistenz-Modus muss bei der Definition des Subsystems
(SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES ...,VERSION-COEXISTENCE=*ALLOWED) verankert und beim Kommando
START-SUBSYSTEM explizit angefordert werden.

Wird bei **temporärer Koexistenz** die Version B eines Subsystems geladen, wenn bereits Version A des Subsystems aktiv ist, werden alle neuen Aufrufer an die Version B angeschlossen. Die Aufträge, die an die Version A angeschlossen sind, werden noch bearbeitet. Nach Bearbeitung aller Aufträge an Version A wird diese automatisch beendet. Bei der Definition ist darauf zu achten, dass die zu ersetzende, „alte“ Version nicht abhängig sein darf von der ersetzenden, „neuen“ Version.

Ist eine **permanente Koexistenz** verschiedener Versionen eines Subsystems gewünscht, muss für jedes dieser Subsysteme das entsprechende Attribut bei der Definition verankert sein und beim Kommando
START-SUBSYSTEM dieser Modus explizit angefordert werden.

3.1.7 Beziehungen zwischen Subsystemen

Als zentrale Instanz verwaltet DSSM die Beziehungen der Subsysteme untereinander, wobei verschiedene Beziehungen explizit überwacht werden können.

Auftragsbeziehungen auf ein Subsystem über expliziten oder impliziten Anschluss werden von DSSM eingerichtet und überwacht. Eine Auftragsbeziehung kann immer zwischen einer Task und einem Auftragseingang geknüpft werden.

Adressbeziehungen über Binder und Lader müssen explizit im Operanden REFERENCED-SUBSYSTEMS der SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES angegeben werden. Der Operand sollte jedoch nur auf Subsysteme der gleichen „Familie“ (Privilegierungsstufe, Speicherklasse usw.) angewandt werden. Bei entkoppelten Subsystemen sollte die Systembetreuung nur dynamische Adressmechanismen wählen (SVC, ISL oder System-Exit-Mechanismen für privilegierte Subsysteme, die BLS-Schnittstelle BIND für nichtprivilegierte Subsysteme).

Adressbeziehungen schränken die Entladbarkeit eines Subsystems ein, da in umgekehrter Reihenfolge als beim Laden entladen werden muss. Adressbeziehungen werden beim Laden und Entladen beachtet, aber nicht überwacht. Sie verhindern das Entladen eines Subsystems unabhängig davon, ob Aufträge bestehen oder nicht.

Abhängigkeitsbeziehungen setzen die Verfügbarkeit eines anderen Subsystems voraus. Diese Beziehungen können mit dem Operanden RELATED-SUBSYSTEMS der SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES deklariert werden und werden bei der Lade- und Entladereihenfolge beachtet.

3.1.8 Beziehungen zwischen Nebenkomponenten und Subsystemen

Während der Installation eines Subsystems wird der Bezug zu seinen Nebenkomponenten hergestellt.

Die **Modulbibliothek** ist eine verpflichtende Nebenkomponente, da kein Subsystem ohne sie geladen werden kann.

Ist eine **Informationsdatei** oder eine **Syntaxdatei** bei der Definition angegeben, erhalten auch diese den Rang einer Pflicht-Komponente, deren Fehlen das Laden des Subsystems verhindert.

Ist eine **Rep-Datei** bei der Definition angegeben, ist diese beim Laden nur dann verpflichtend, wenn dies bei der Definition explizit vereinbart wird (REP-FILE-MANDATORY=*YES).

Eine **Meldungsdatei** kann bei der Definition angegeben werden, ist aber keine Pflicht-Komponente, da das Laden des Subsystems trotz fehlender Meldungsdatei durchgeführt werden kann.

In folgenden Situationen sendet DSSM eine Frage an die Bedienstation, die vom Operator beanwortet werden muss:

- die Modulbibliothek eines verpflichtenden Subsystems fehlt (Attribute *AT-DSSM-LOAD oder *MANDATORY-AT-STARTUP)
- die angegebene Informationsdatei ist zwar deklariert aber nicht vorhanden
- die Rep-Datei mit dem Attribut REP-FILE-MANDATORY=*YES fehlt

Der Operator hat trotz einer fehlenden Datei die Möglichkeit, den Startvorgang fortzusetzen (z.B. das Subsystem ohne Informationsdatei zu starten). Die Verantwortung für eine mögliche abnormale Subsystem- oder Systembeendigung liegt ausschließlich beim Operator.

3.1.9 Kommunikation zwischen Subsystem und DSSM

Der Austausch von Informationen und Meldungen ist erforderlich bei subsystemspezifischen Routinen zur Initialisierung, Deinitialisierung, Auftragsbeendigung und Kontrolle der Auftragsbeendigung (INIT-, DEINIT-, STOPCOM- und CLOSE-CTRL-Routinen).

Der Kommunikationsbereich besteht aus einem Informationsbereich für die gestartete Routine (DSSM --> Subsystem) und einem Rückmeldungsbereich (Subsystem --> DSSM).

Das Starten der Routine erfolgt über

1. Prozeduraufruf
   bei Initialisierung (ohne Bedingung) oder bei Deinitialisierung, Auftragsbeendigung und Kontrolle der Auftragsbeendigung, wenn die Holdertask nicht als Arbeitstask genutzt wird
2. die bei der Initialisierung übermittelte Schnittstelle (Börse oder FITC) wenn die Holdertask als Arbeitstask genutzt wird

Die Rückmeldung an DSSM erfolgt über Prozedur-Rücksprung mit Rückmeldungsbereich bei Deinitialisierung, Auftragsbeendigung und Kontrolle der Auftragsbeendigung, wenn die Holdertasks nicht oder nicht mehr als Arbeitstask verwendet wird. Wird die Holdertask als Arbeitstask genutzt, erfolgt die DSSM-Benachrichtigung mittels NOTIFY-Aufruf mit dem Rückmeldungsbereich als Input-Parameter.

3.1.10 Informieren über Subsysteme

Der Benutzer kann über die Kommandos SHOW-SUBSYSTEM-STATUS und SHOW-SUBSYSTEM-INFO den Zustand eines Subsystems abfragen. Darüber hinaus kann er sich einen Überblick über die gesamte Subsystem-Konfiguration verschaffen.

3.1.11 Zustände eines Subsystems

NOT-CREATED

Das Subsystem ist im laufenden System deklariert, wurde aber bisher noch nicht durch ein START-SUBSYSTEM-Kommando aktiviert, oder es ist nach vorhergehender Aktivierung wieder deaktiviert. Eine Task kann auf dieses Subsystem nicht zugreifen, bevor es aktiviert ist.

IN-CREATE

Das Subsystem wird aktiviert. Der Lade- und Initialisierungsprozess ist noch nicht abgeschlossen. Eine Task kann auf dieses Subsystem noch nicht zugreifen.

CREATED

Das Subsystem ist geladen und initialisiert. Tasks können auf das Subsystem zugreifen.

IN-DELETE

Das Subsystem ist durch ein STOP-SUBSYSTEM-Kommando deaktiviert. Der Entlade- und Deinitialisierungsprozess ist noch nicht abgeschlossen. Eine weitere Task kann auf dieses Subsystem nicht mehr zugreifen.
An das Subsystem angeschlossene Tasks werden noch abgearbeitet.

IN-HOLD

Das Subsystem ist durch ein HOLD-SUBSYSTEM-Kommando angehalten. Der Deinitialisierungsprozess ist noch nicht abgeschlossen.
Weitere Tasks können auf dieses Subsystem nicht mehr zugreifen. An das Subsystem angeschlossene Tasks werden noch abgearbeitet.

IN-RESUME

Das Subsystem wird durch ein RESUME-SUBSYSTEM-Kommando fortgesetzt. Der Reinitialisierungsprozess ist noch nicht abgeschlossen. Eine Task kann auf dieses Subsystem noch nicht zugreifen.

NOT-RESUMED

Das Subsystem ist durch ein HOLD-SUBSYSTEM-Kommando angehalten. Der Deinitialisierungsprozess ist abgeschlossen.
Keine Task kann auf dieses Subsystem zugreifen, bis ein RESUME-SUBSYSTEM-Kommando angegeben und dessen Ausführung abgeschlossen ist.

LOCKED

Ein nicht behebbarer Fehler ist aufgetreten, während das Subsystem aktiv war bzw. aktiviert, deaktiviert, fortgesetzt oder angehalten wurde. Ein weiterer Versuch, die entsprechenden Kommandos abzusetzen, wird abgewiesen.

Folgende Situationen können ein Subsystem in den Zustand LOCKED überführen:

- Der Übergang in den Zustand LOCKED ist in der INIT-, DEINIT, STOPCOM-, CLOSE-CTRL-Routine verankert (gilt nur für privilegierte Subsysteme).
- Die Holdertask des Subsystems ist abnormal beendet worden und ein Restart für diese Task ist entweder nicht vorgesehen oder nicht durchführbar (siehe "Fehlerbehandlung in DSSM").

- Während des nicht-unterbrechungsfreien Austauschs von Versionen tritt ein Problem beim Deaktivieren der alten Version auf. Unabhängig vom Wert des Operanden RESTART ist das Subsystem im Zustand LOCKED; die Aktivierung der neuen Version wird fortgesetzt.

Beim Aktivieren, Deaktivieren, Anhalten und Fortsetzen eines Subsystems ändert sich der Zustand des Subsystems und erreicht (ausgehend von einem Anfangszustand), einen Zielzustand; z.B. ist der Zielzustand beim Aktivieren des Subsystems der Zustand CREATED.

Es gibt verschiedene Anfangszustände für eine Anforderung; z.B. ist das Kommando START-SUBSYSTEM für ein Subsystem akzeptabel, dessen Zustand NOT-CREATED oder IN-DELETE ist, wenn der Operand RESET=*YES gesetzt ist.

Alle Zustände, die ein Subsystem annehmen kann, sind in der folgenden Tabelle zusammengefasst. Ein Zustandswechsel, der durch ein DSSM-Kommando erreicht wird, ist in einer Zeile aufgezeigt. Der Anfangszustand des Subsystems ist mit 1 bezeichnet. Die höchste Ziffer in der Zeile markiert den Zielzustand, der durch das entsprechende DSSM-Kommando erreicht werden kann. Mögliche Zwischenzustände sind ebenfalls aufgezeigt.

DSSM-Kommandos	**Zustände eines Subsystems**							**Operand des DSSM-Kommandos** [1]
	NOT-CREATED	IN-CREATE	CREATED	IN-DELETE	IN-HOLD	IN-RESUME	NOT-RESUMED	
START-SUBSYSTEM	1	2	3					---
			3			2	1	---
	1	2	3					RESET=*YES
		2	3	1				RESET=*YES
			3		1	2		RESET=*YES
			3			2	1	RESET=*YES
STOP-SUBSYSTEM	3		1	2				---
	3			2			1	---
	2				1			FORCED=*YES
	2			1				FORCED=*YES
HOLD-SUBSYSTEM			1		2		3	---
					1		2	FORCED=*YES
RESUME-SUBSYSTEM			3			2	1	---
			3			2	1	RESET=*YES
			3		1	2		RESET=*YES

[1]Operand des DSSM-Kommandos, der für den Zustandswechsel gesetzt werden muss

Tabelle 4: Zustände eines Subsystems

3.1.12 Subsystem-Überwachung mittels Monitor-Jobvariablen

Subsysteme können mittels Monitor-Jobvariablen (MONJV) überwacht werden. Die MONJV muss im Kommando START-SUBSYSTEM angegeben werden. DSSM verwaltet und setzt die MONJV während der gesamten Laufzeit des Subsystems bis zu seiner Beendigung bei:

- expliziten DSSM-Aufrufen (HOLD-SUBSYSTEM, /RESUME-SUBSYSTEM, /STOP-SUBSYSTEM, /UNLOCK-SUBSYSTEM)
- impliziten Operationen (Subsystem, automatische Wiederherstellung ...)
- SHUTDOWN

Die MONJV zeigt an, ob das Subsystem aktiv, angehalten, unterbrochen oder gesperrt ist. Die MONJV kann folgende Inhalte haben:

Byte	Länge	Inhalt	Werte
1	3	Status	$R (running) / $A (abnormal end) / $L (loaded) / $T (terminate)
4	1	reserviert	0
5	4	TSN	???? *(4 Fragezeichen)*
9	4	Home Catid	
13	4	reserviert	
17	3	Session-Nummer	
20	51	Zeitstempel	
71	3	Session-Nummer	
74	8	Name des Subsystems	
82	7	Version des Subsystems	
89	15	Zustand des Subsystems	für $R: created für $A: abnormal end / locked für $L: in create für $T: not created / not resumed / in delete / in resume / in hold
104	24	unbenutzt	
128	127	reserviert für Anwender des Subsystems	

Tabelle 5: Inhalte der Monitor-Jobvariablen

3.1.13 Funktionsübersicht

Die Tabelle gibt einen Überblick über die von DSSM/SSCM angebotenen Funktionen und welche Operanden für die einzelnen Subsystem-Klassen angegeben werden können.

Funktion (Operanden)	**TPR SAR**	**TU SAR**	**TU BAR**	**Sys-Exits**	**Share Prod.**	**PK**
Subsystem-Deklaration mit der SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES						
Identifikation						
SUBSYSTEM-NAME, VERSION,-DYNAMIC-CHECK-ENTRY	X	X	X	X	X	
Binden und Laden						
LIBRARY, REP-FILE, LINK-ENTRY, AUTOLINK,-UNRESOLVED-EXTERNALS	X	X	X	X	X	
CREATION-TIME=*BEFORE-DSSM-LOAD/ *AT-DSSM-LOAD/*BEFORE-SYSTEM-READY	X			X		
CREATION-TIME=*AFTER-SYSTEM-READY/ *AT-CREATION-REQUEST	X	X	X	X	X	
CREATION-TIME=*AT-SUBSYSTEM-CALL	1					
REFERENCED-SUBSYSTEMS	2	2	2	2	2	
Adressraum bestimmen						
MEMORY-CLASS=*SYSTEM-GLOBAL	X	X		X	10	
MEMORY-CLASS=*=*LOCAL-PRIVILEGED			X		10	
MEMORY-CLASS=*=*LOCAL-UNPRIVILEGED			X		10	
MEMORY-CLASS=*=*BY-SLICE		X	X		10	
SUBSYSTEM-ACCESS=*SYSTEM	X			X		
SUBSYSTEM-ACCESS=*=*HIGH		X	X		X	
SIZE			X		X	
START-ADDRESS			3		3	
Nebenkomponenten						
MESSAGE-FILE, SYNTAX-FILE	X	X	X	8	8	
SUBSYSTEM-INFO-FILE	X			8		
Starten und Beenden						

INIT-, STOPCOM-, DEINIT-, CLOSE-CTRL-Routine,- INTERFACE-VERSION	X			X		
Holdertask zum Ablauf						
RESTART-REQUIRED	X			X		
Ablauf (Operand SUBSYSTEM-ENTRIES=(...))						
MODE=*LINK	X	X	X		X	
MODE=*SVC/*SYS-EXIT/*ISL	X			X		
CONNECTION-ACCESS=*SYSTEM	X			X		
CONNECTION-ACCESS=*ALL		X	X		X	
CONNECTION-SCOPE=*PROGRAM/*TASK/*FREE	X	X	X	X	X	
CONNECTION-SCOPE=*CALL/*OPTIMAL	9					
Subsystem-Konfiguration mit den SSCM-Anweisungen						
ASSIGN-HOLDER-TASK	X	4	X		8	
SET-SUBSYSTEM-ATTRIBUTES (Operand RELATED-SUBSYSTEMS)	X	X	X	X	X	
SET-SUBSYSTEM-ATTRIBUTES (Operand REFERENCED-SUBSYSTEMS)	11	11	11	11	11	
SEPARATE-ADDRESS-SPACE			5		5	
Subsystem-Konfiguration mit den DSSM-Kommandos						
START-SUBSYSTEM, STOP-SUBSYSTEM,- HOLD-SUBSYSTEM, RESUME-SUBSYSTEM	X	X	X	X	X	
ADD-/REMOVE-/UNLOCK-SUBSYSTEM,- SAVE-SUBSYSTEM-CATALOG						X
MODIFY-SUBSYSTEM-PARAMETER	X	X	X	X	X	X
Subsystem-Information mit den DSSM-Kommandos						
SHOW-SUBSYSTEM-ATTRIBUTES						X
SHOW-SUBSYSTEM-INFO						X
SHOW-SUBSYSTEM-STATUS	X	X	X	X	X	
globale Subsystem-Verwaltung mit den DSSM-Kommandos						
RELEASE-SUBSYSTEM-SPACE			6			
SET-DSSM-OPTIONS	7	7	7	7	7	7

Tabelle 6: DSSM-Funktionsübersicht

Erläuterung

TPR SAR	privilegierte Subsysteme (nur Systemadressraum)
TU SAR	nicht-privilegierte Subsysteme mit Systemadressraum
TU BAR	nicht-privilegierte Subsysteme mit Benutzeradressraum
Sys-Exits	relevant für System-Exits
Share-Prod.	relevant für Share-Produkte
PK	Privilegierte Benutzerkennung für die Systembetreuung (Kennung mit dem Privileg SUBSYSTEM-MANAGEMENT)
X	Funktion verfügbar
1	nur für Subsysteme mit SVC- und/oder ISL-Anschluss
2	Beziehungen nur von BAR nach SAR
3	nur für Subsysteme im Klasse-6-Speicher
4	im Systemadressraum ist die Holdertask nur für Abläufe erforderlich
5	nur für Subsysteme im Adressraum-Streifen nötig
6	Reservierung für den nicht-privilegierten bzw. beide Adressraum-Streifen kann aufgehoben werden
7	nur für Diagnose und Test
8	nur soweit sinnvoll
9	nur in Verbindung mit MODE=*SVC bzw. MODE=*ISL
10	je nach Ablage im Adressraum
11	sollte für Subsysteme der gleichen „Familie“ reserviert werden (Privilegierung, Speicherklasse usw.)

3.2 Speicher- und Taskkonzept

Speicherkonzept

Subsysteme werden je nach Deklaration folgendermaßen geladen:

- in den Systemadressraum (Klasse-3- oder Klasse-4-Speicher)
  bei MEMORY-CLASS=*SYSTEM-GLOBAL
- in den Benutzeradressraum (Memory Pool: Klasse-5- oder Klasse-6-Speicher) bei
  MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED
- in beide Adressräume bei MEMORY-CLASS=*BY-SLICE

Anschlüsse an Subsysteme im Benutzeradressraum sind nur möglich, wenn der zugewiesene Adressraum auch in der eigenen Task frei ist. Der Code muss nach dem Anschluss immer an der gleichen Adresse liegen. Für allgemein verfügbare Subsysteme im Benutzeradressraum, die jederzeit und von allen Tasks angesprochen werden können, wird ein fester Bereich im Klasse-5-Speicher reserviert, der so genannte Adressraum-Streifen. Diesen Adressraum-Streifen teilen sich alle allgemein verfügbaren Subsysteme, wobei für nicht-privilegierte Subsysteme je ein eigener Streifen zur Verfügung steht.

Für Benutzertasks können bei nicht ausreichendem Klasse-5-Speicher Probleme auftreten. Der für Subsysteme reservierte Adressraum-Streifen kann durch das DSSM-Kommando RELEASE-SUBSYSTEM-SPACE freigegeben werden, um mehr Klasse-5-Speicher zur Verfügung zu haben.

Subsysteme, die nicht gleichzeitig gebraucht werden (die sich also nicht gegenseitig aufrufen), können im Adressraum parallel liegen. Je mehr Subsysteme parallel liegen, umso kleiner ist der benötigte Adressraum-Streifen. Dabei ist auf die Ausgewogenheit der parallelen Subsysteme zu achten, da hohe Parallelität einerseits eine Einsparung des Adressraums, andererseits eine mögliche Verschlechterung der Performance zur Folge haben kann (je mehr Parallelität, umso mehr gegenseitige Verdrängung). Die Verteilung der Subsysteme im Adressraum-Streifen kann mit der SSCM-Anweisung SEPARATE-ADDRESS-SPACE geregelt werden.

Adressraum-Haushalt

Der Adressraum-Haushalt von DSSM gibt die Möglichkeit, den Systemadressraum (Klasse-3- und Klasse-4-Speicher) zu entlasten, indem Subsysteme in den Benutzeradressraum der Holdertask gelegt werden. Das ist jedoch nur für nichtprivilegierte Subsysteme relevant, denn alle privilegierten Subsysteme werden immer in den Systemadressraum geladen (MEMORY-CLASS=*SYSTEM-GLOBAL).
Die Verlagerung ist dann problemlos, wenn die Subsysteme sich nicht gegenseitig aufrufen und nicht über ein drittes Programm voneinander abhängen oder im Klasse-6-Speicher als Hauptprogramm ablaufen können.
Zu beachten ist, dass nur die Verlagerung nicht-privilegierter Subsysteme angestrebt werden kann, die unterhalb der 16-MByte-Grenze angesiedelt sind.
Für alle anderen nicht-privilegierten Subsysteme wird das Laden oberhalb der 16-MByte-Grenze empfohlen (SUBSYSTEM-ACCESS=*HIGH).

Auch Subsysteme, die reentrant-fähig sind und als Hauptprogramme laufen, können aus dem Systemadressraum ausgelagert werden. Sie müssen durch MEMORY-CLASS=*LOCAL-UNPRIVILEGED im Memory Pool im Klasse-6-Speicher verfügbar sein.

Für Subsysteme, die aus einem mehrbenutzbaren Code (Programmbereich) und einem nicht-mehrbenutzbaren Code (Datenbereich) bestehen, gibt es das Konzept zur Minimierung der Systemadressraum-Belegung:
Der Programmbereich wird in den gemeinsam benutzbaren Adressraum geladen (das entspricht MEMORY-CLASS=*SYSTEM-GLOBAL). Der Datenbereich wird in den Benutzeradressraum der Holdertask geladen und zum Zeitpunkt des Anschlusses einer Task an das Subsystem in deren privaten Benutzeradressraum an dieselbe Adresse kopiert.
Dieses Konzept wird mit der Definition eines Subsystems mit MEMORY-CLASS=*BY-SLICE realisiert.

Die Vorteile dieses Konzepts sind:

- Adressbezogene Referenzen zwischen Programm- und Datenbereich sind immer möglich, weil die Kopie des Datenbereichs an derselben Adresse beginnt wie das Original.
- Die Performance wird durch diese Art der Adressraumaufteilung wesentlich erhöht, weil beim Anschluss einer Task an das Subsystem kein Zugriff zur Programm- oder Bindemodulbibliothek erfolgt und Externverweise nicht aufgelöst werden müssen.

Die Nachteile dieses Konzepts sind:

- Der zum Startup-Zeitpunkt bestimmte und vorreservierte Adressraum für den Datenbereich kann nur sehr begrenzt erweitert werden. Eine Speicherplatzoptimierung ist wegen der strengen Adressraumaufteilung nicht möglich.
- Ist der zur Aufnahme des Datenbereichs vorgesehene Adressraum der sich anschließenden Task bereits belegt, wird der Subsystemcode (Daten- und Programmbereich!) vollständig in den Benutzeradressraum dieser Task geladen.

Die Verlagerung vom Systemadressraum in den Benutzeradressraum lohnt sich nur in folgenden Fällen:

1. Subsysteme, die voneinander unabhängig sind und von Benutzerprogrammen nicht gleichzeitig benutzt werden, können parallel in den Benutzeradressraum gelegt werden. Ist dies nicht der Fall, ist eine Verlagerung nicht sinnvoll, da der Verbrauch im Benutzeradressraum ungünstiger ausfällt als im Systemadressraum.
2. Die Subsysteme sind ausreichend groß, um den auftretenden Verschnitt durch die Verwendung von Memory Pool auszugleichen (min. Größe eines Memory Pools: 1 MByte).
3. Das Subsystem muss unterhalb der 16-MByte-Grenze angesiedelt sein. (Für Subsysteme, die auch oberhalb der Grenze geladen werden dürfen, kann der Operand SUBSYSTEM-ACCESS=*HIGH verwendet werden, um einer Überlastung der unteren 16-MByte zu begegnen.)

Zusammenfassung:
Unabhängige Subsysteme parallel in Memory Pools zu konfigurieren ist nur dann sinnvoll, wenn die Summe der Größe aller Subsysteme mehr als 1 MByte beträgt und kein Subsystem die 1 MByte-Grenze in seiner Größe überschreitet.

Taskkonzept - Holdertask

Die Aktivierung eines Subsystems erfolgt unter einer eigenen Task, der Holdertask. Abhängig vom Typ des Subsystems kann diese Task als Subsystem-Arbeitstask oder nur als reine Holdertask verwendet werden. Der Benutzeradressraum dieser Task kann für die Auslagerung aus dem Systemadressraum genutzt werden.

Die Anzahl der benötigten Holdertasks sollte möglichst gering gehalten werden. Eine hohe Taskanzahl beeinflusst zwar die Parallelität günstig, da umso mehr Subsysteme installiert werden können, je mehr Tasks gleichzeitig angelegt werden. Andererseits benötigen mehr Tasks auch mehr taskspezifische Tabellen.

DSSM selbst minimiert die Anzahl der Holdertasks; die Verteilung der Subsysteme kann aber durch die SSCM-Anweisung ASSIGN-HOLDER-TASK (siehe "ASSIGN-HOLDER-TASK") bei der Generierung des Subsystemkatalogs mit SSCM gesteuert werden.
Standardmäßig werden alle Subsysteme, die mit MEMORY-CLASS=*BY-SLICE definiert wurden, an dieselbe Holdertask angeschlossen.

Im Fehlerfall wird automatisch der Wiederanlauf der Holdertask initiiert, um nicht alle Subsysteme, die an die Holdertask angeschlossen sind, in Mitleidenschaft zu ziehen. Gesteuert durch den Operanden RESTART-REQUIRED (Anweisung SET-SUBSYSTEM-ATTRIBUTES) ist auch der Wiederanlauf eines Subsystems vorgesehen. Der Operand ermöglicht, die Subsystem-Initialisierung dann noch einmal aufzurufen, wenn die Holdertask während des Ablaufs von Subsystem-Routinen beendet wurde (siehe "Fehlerbehandlung in DSSM").

3.3 Verwaltung von Share-Programmen

DSSM unterstützt die Verwaltung von Share-Programmen. Dies ermöglicht im Systemlauf:

- das Entladen von Share-Programmen auch aus dem Klasse-4-Speicher
- das Auslagern von Share-Programmen vom Klasse-4-Speicher in den Klasse-5- bzw. Klasse-6-Speicher

Share-Programme können als Subsysteme deklariert und als solche von DSSM verwaltet werden. Sie können also wie andere Subsysteme dynamisch aktiviert, deaktiviert, angehalten und fortgesetzt werden.

Voraussetzung ist, dass die Share-Programme bei der Generierung des Subsystemkataloges deklariert worden sind. Mit dieser Definition kann nach erfolgter Inbetriebnahme des DSSM ein Share-Programm ebenso aktiviert und deaktiviert werden wie ein Subsystem.

Die Benutzer können auf das Programm über den BIND-Makro bzw. mit den Kommandos START-EXECUTABLE-PROGRAM und LOAD-EXECUTABLE-PROGRAM zugreifen.

Der Einsprungspunkt muss immer mit der SSCM-Anweisung SET-SUBSYSTEM-ATTRIBUTES,....SUBSYSTEM-ENTRIES= angegeben werden, auch wenn er mit dem LINK-ENTRY (der für das Laden des Subsystems verwendete Bezugsadresse) identisch ist.

3.4 Verwaltung des dynamischen Subsystemkatalogs

Erweitern des dynamischen Subsystemkatalogs

Während des Systemlaufs kann die aktuelle dynamische Subsystem-Konfiguration von der Systembetreuung erweitert werden (Kommando ADD-SUBSYSTEM). Der Katalog für die neue Subsystem-Konfiguration muss mit SSCM generiert werden und es müssen folgende Punkte beachtet werden:

- Der neue Katalog kann wahlweise größer als der alte sein (d.h. auch alle Subsysteme des alten Kataloges enthalten) oder als „Delta“-Katalog angelegt sein und lediglich die neuen Definitionen von Subsystemen enthalten.
- Der „alte“ Subsystemkatalog, der während Startup benutzt wurde, wird nicht automatisch aktualisiert. Beim nächsten Startup ergeben sich folgende Möglichkeiten:
  - der „neue“ Katalog wird benutzt
  - ein ganz neu erstellter Katalog wird benutzt, in dem z.B. die nicht mehr benutzten Versionen von Subsystemen nicht mehr enthalten und Änderungen von Eigenschaften (wie z.B. CREATION-TIME) bereits vorgenommen sind
- Anweisung ASSIGN-HOLDER-TASK

  Die Anweisung darf nicht für alte und neue Subsysteme gegeben werden.

  Beispiel

Subsysteme im alten Katalog: A, B, C Subsysteme im neuen Katalog: A, B, C, D und E

dann ist erlaubt:
`//ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(A,B))`
`//ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(D,E))`

dann ist nicht erlaubt: `//ASSIGN-HOLDER-TASK TYPE=SHARED-HOLDER(BY-SUBSYSTEMS=(A,B,E))`

- Anweisung SET-SUBSYSTEM-ATTRIBUTES

  Der Operand CREATION-TIME für ein neues Subsystem muss verträglich sein mit den bereits existierenden Versionen des Subsystems in der alten Subsystem-Konfiguration.

  Beispiel

  Subsystem X in der Version 1 ist im alten Katalog deklariert mit CREATION-TIME=*AT-SUBSYSTEM-CALL. Dann kann Subsystem X in der Version 2 im neuen Katalog nur CREATION-TIME=*AT-CREATION-REQUEST haben.

  Falls für ein neues Subsystem CREATION-TIME=*BEFORE-SYSTEM-READY oder *AFTER-SYSTEM-READY ist, dann wird das Subsystem nicht kreiert, da der Zeitpunkt „SYSTEM READY“ (Systemeinleitung) nicht mehr gegeben ist. Es erfolgt eine entsprechende Meldung.

  Neu hinzuzufügende Subsysteme mit den Attributen MEMORY-CLASS=*LOCAL-PRIVILEGED (Klasse-5-Speicher) dürfen weder die Breite des System- bzw. Benutzer-Adressraum-Streifens noch die Lage der alten Subsysteme in diesem Streifen ändern.

  Es dürfen keine Abhängigkeiten von alten zu neuen Subsystemen deklariert werden (Operanden REFERENCED-SUBSYSTEM und RELATED-SUBSYSTEM).

  Beispiel

Subsysteme im alten Katalog: A, B, C Subsysteme im neuen Katalog: A, B, C, D und E	
Wenn X --> Y bedeutet, dass X zur Auflösung von Extern-Aufrufen Y benötigt,	
dann ist	A --> D nicht erlaubt, D --> E erlaubt, D --> B,C erlaubt.

- Anweisung SEPARATE-ADDRESS-SPACE

  Neue und alte Subsysteme dürfen disjunkt sein, sich also adressmäßig überschneiden.

  Beispiel

Subsysteme im alten Katalog: A, B, C Subsysteme im neuen Katalog: A, B, C, D und E	
Wenn X --> Y bedeutet, dass X zu Y disjunkt ist,	
dann ist	A --> D erlaubt, D --> E erlaubt, D --> B,C erlaubt

Dynamische Änderungen von Subsystem-Attributen

Mit dem Kommando MODIFY-SUBSYSTEM-PARAMETER ist es möglich, Eigenschaften von Subsystemen dynamisch zu ändern. Es ist weder ein Shutdown noch das Hinzufügen einer neuen Version des Subsystems nötig. Mit dem Kommando SAVE-SUBSYSTEM-CATALOG werden die Konfigurationsänderungen für den nächsten Startup abgespeichert.

Diese Möglichkeiten sind vor allem in folgenden Situationen hilfreich:

- Um ein Subsystem stoppen zu können, müssen alle angeschlossenen Tasks ihre Verbindung abgebaut haben. Wurde das Subsystem nicht mit dem Attribut FORCED-STATE-CHANGE=*ALLOWED definiert, besteht keine Möglichkeit, das Deaktivieren des Subsystems zu beschleunigen. Ist dies aber aus bestimmten Gründen nötig, kann mit dem Kommando MODIFY-SUBSYSTEM-PARAMETER das Attribut dynamisch geändert werden.
- Kommt es zu einer Memory-Pool-Kollision, weil z.B. zwei Subsysteme an derselben Adresse liegen und die Task nur eins von beiden nutzen kann, ist es mit dem Kommando MODIFY-SUBSYSTEM-PARAMETER möglich, die Adressen der Subsysteme zu ändern und so die Memory-Pool-Kollision zu vermeiden.

3.5 Inbetriebnahme der dynamischen Subsystemverwaltung

Die dynamische Subsystemverwaltung wird während der BS2000-Systemeinleitung gestartet.

Über den Parameterservice werden alle zur Initialisierung von DSSM notwendigen Informationen mitgegeben. Dazu zählen der Name des statischen Subsystemkatalogs und die DSSM-Versionsnummer. Des Weiteren kann bereits - wenn unbedingt erforderlich - die Protokollierung von DSSM-spezifischen Daten für die Fehlerdiagnose eingeschaltet werden.

Alle Sätze, die über den Parameterservice verarbeitet wurden, werden in Form von Meldungen in der Protokolldatei CONSLOG erfasst.

Das Schlüsselwort für die Inbetriebnahme der Subsystemverwaltung ist **DSSM**.

Der Ablauf der Inbetriebnahme von DSSM über den Parameterservice wird auch im Handbuch „Einführung in die Systembetreuung“ beschrieben.

Für den späteren Betrieb von DSSM V21.0 werden folgende Dateien unter der Benutzerkennung TSOS auf dem Home-Pubset vorausgesetzt:

SYSLNK.DSSM.210	Bibliothek mit Lademodulen für DSSM
SKMLNK.DSSM.210	Bibliothek mit Lademodulen für DSSM x86-Variante
SYSNRF.DSSM.210	Verweisdatei für DSSM-Rep-Datei-Verarbeitung (NOREF-Datei)
SYSREP.DSSM.210	Rep-Korrekturdatei für DSSM
SYSSDF.DSSM.210	SDF-Syntaxdatei
SYSMES.DSSM.210	Meldungsdatei

Der zu erzeugende Subsystemkatalog muss ebenfalls auf den Home-Pubset gebracht und unter der Benutzerkennung TSOS abgelegt werden. Der Name des Katalogs ist frei wählbar und kann über den Parameterservice dem System bekannt gemacht werden.

Für die Generierung eines Subsystemkatalogs steht SSCM zur Verfügung, das im Kapitel SSCM ausführlich beschrieben wird.

Problembehandlung beim Systemlauf

Wenn DSSM nicht initialisiert werden kann, wird in einer Meldung der Grund dafür angezeigt (z.B. fehlender statischer Subsystemkatalog) und die Meldung `ESM0401` ausgegeben. Der Operator kann während des Systemlaufs an der Bedienstation einen neuen statischen Subsystemkatalog angeben, wenn er in der Parameterdatei nicht festgelegt wurde oder der Standard-Katalog (SYS.SSD.CAT.X) nicht vorhanden ist.

Der Systemlauf wird im Allgemeinen nicht fortgesetzt, wenn verpflichtende Subsysteme nicht in Betrieb genommen werden können. Der Grund für den Fehler bei der Initialisierung von DSSM muss erst behoben werden. Anschließend ist der Systemlauf zu wiederholen.

Wenn Subsysteme mit dem Attribut *BEFORE-SYSTEM-READY nicht in Betrieb genommen werden können, setzt DSSM den Systemlauf fort.

Wenn eine der folgenden Dateien eines Subsystems mit den Attributen *AT-DSSM-LOAD oder *MANDATORY-AT-STARTUP fehlt, kann der Operator während des Systemlaufs einen neuen, gültigen Dateinamen angeben:

- Informationsdatei
- Rep-Korrekturdatei
- Modulbibliothek

Wird vom Operator kein neuer Dateiname für die fehlende Rep-Korrekturdatei oder die Modulbibliothek eingegeben, wird der Systemlauf gestoppt.

Wenn die Informationsdatei fehlt, kann der Operator

- einen neuen, gültigen Dateinamen eingeben
- die Meldung ignorieren und den Systemlauf fortsetzen
- die Meldung ignorieren und den Systemlauf stoppen.

Format des Parametersatzes zur Inbetriebnahme von DSSM

Format	Bedeutung
SSMCAT=name	Name des statischen Subsystemkatalogs
VERSION=versnr	Versionsnummer von DSSM
LOGGING=ON / OFF	Steuert die DSSM-spezifische Protokollierung zur Fehlerdiagnose
REPFILE=<repfile name>	Name der REP-Korrekturdatei für das Laden von DSSM

Für Subsysteme, die nicht während der Systemeinleitung automatisch aufgebaut werden, müssen im BS2000-Systemlauf START-SUBSYSTEM-Kommandos gegeben werden.

Ausschnitt aus der Parameterdatei

```
/BS2000 PARAMS
:
/BEGIN DSSM
 SSMCAT=name ————————————————————————————————————————————  (1)
 VERSION=versnr ——————————————————————————————————————————  (2)
 LOGGING=ON / OFF ————————————————————————————————————————  (3)
 REPFILE=repfile name————————————————————————————————————  (4)
/EOF
:
/END-PARAMS
```

(1) Die Steuer- und Parametersätze müssen in der Parameterdatei nur vorhanden sein, wenn die Systembetreuung von folgenden Standardwerten abweichen will:

für BS2000/OSD-BC V5.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=040
für BS2000/OSD-BC V4.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=040 oder SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=039 oder SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=038
für BS2000/OSD-BC V3.0:	SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=040 oder SSMCAT=$TSOS.SYS.SSD.CAT.X und VERSION=036

(2) Die Versionsnummer bezieht sich auf alle DSSM-spezifischen Dateinamen(z.B. SYSLNK.DSSM.040, SYSREP.DSSM.040 bei BS2000/OSD-BC V5.0).

(3) Die Anweisung LOGGING=OFF schaltet die Protokollierung aus. Mit LOGGING=ON wird bereits während der Inbetriebnahme von DSSM ein Protokoll der Diagnosedaten erzeugt. Wenn nur Standardwerte angegeben werden, ist das LOGGING nicht möglich.

(4) Name der gewünschten REP-Korrekturdatei für das Laden von DSSM. Ist der Parameter nicht angegeben, wird DSSM mit dem Standardnamen der Rep-Korrekturdatei geladen (SYSREP.DSSM.*version*).

3.6 Abrechnungssätze des DSSM (Accounting)

Allgemeines zum Abrechnungssystem des BS2000

Das gesamte Abrechnungssystem des BS2000 wird von der Systembetreuung gesteuert. Diese bestimmt den Zeitpunkt, an dem das Abrechnungssystem gestartet werden soll, vereinbart den Namen der Abrechnungsdatei und legt Namen und Anzahl der Abrechnungssätze und Satzerweiterungen fest, die in der Abrechnungsdatei erfasst werden sollen.
Die Systembetreuung legt ferner den Zyklus und den Umfang der periodischen Erfassung fest, die bestimmte Abrechnungsätze und Jobklassen umfasst.

Mit dem Abrechnungssystem kann die Systembetreuung dynamisch Abrechnungssätze ganz oder teilweise ein- und ausschalten und Einfluss auf den Umfang der einzelnen Abrechnungssätze nehmen.
Mit dem Kommando MODIFY-ACCOUNTING-PARAMETERS können nicht benötigte Abrechnungssätze und Satzerweiterungen ausgeschaltet werden.

Die bei der Darstellung pro Datenfeld verwendeten Attribute sind:

Feld	laufende Nummer des Datenfeldes innerhalb des beschriebenen Satzteils
Distanz	relativer Abstand des Datenfeldes zum Anfang des beschriebenen Satzteils
Länge	Länge des Datenfeldes in Byte
Format	A = alphanumerisch B = Binärzahl B2 = binäre Darstellung der CPU-Zeit C = abdruckbare Zeichen, inklusive Sonderzeichen F = Dateiname X = nicht abdruckbare Zeichen Z = ungepackte Dezimalzahl (*) * = wird bei den einzelnen Satzarten oder Erweiterungselementen festgelegt - = reserviert für künftige Erweiterungen, enthält entweder Leerzeichen oder binär null
(*)	die Uhrzeit wird in der Form `hhmmss`, das Datum in der Form `jjmmtt` dargestellt

Ein Abrechnungssatz besteht aus folgenden vier Teilen:

(A) Satzbeschreibung	Satzkennung, Zeitstempel, ...
(B) Kennzeichnungsteil	Benutzerkennung, Abrechnungsnummer,überwachte Betriebsmittel, ...
(C) Grundinformation	Standarddaten
(D) Variable Information	Satzerweiterungen

Für nähere Informationen zum Abrechnungssystem des BS2000 siehe Handbuch „Einführung in die Systembetreuung“.

Subsystem-Satzbeschreibung

Die Satzbeschreibung (A) enthält eine Satzkennung, die die einzelnen Abrechnungssätze voneinander abgrenzt, einen Zeitstempel sowie Angaben zur Länge des Kennzeichnungsteils und der Grundinformation.

Adresse der Satzbeschreibung = Satzanfang

Aufbau und Inhalt:

Feld-Nr.	Distanz		Länge	Format	Bedeutung
	hex	dez	(Byte)		
1	00	0	4	A	Satzkennung [1]
2	04	4	8	B	Zeitstempel der Tageszeituhr [2]
3	0C	12	2	B	Länge des Kennzeichnungsteils
4	0E	14	2	B	Länge der Grundinformation
5	10	16	4	-	- reserviert -

[1]Die Satzkennung dient zur Unterscheidung der einzelnen Satzarten.

[2]Der Zeitstempel wird vom System als letzte Information in den Satz eingetragen, nachdem der Satz fertig erstellt bzw. dem Abrechnungssystem übergeben wurde.

Länge der Satzbeschreibung: 20 Byte

Subsystem-Kennzeichnung

Die Subsystem-Kennzeichnung im Kennzeichnungsteil (B) beschreibt das Subsystem, auf das sich die Subsystem-Abrechnungsdaten beziehen.

Aufbau und Inhalt:

Feld-Nr.	Distanz		Länge	Format	Bedeutung
	hex	dez	(Byte)		
1	00	0	8	A	Name des Subsystems
2	08	8	7	A	Version des Subsystems
3	0F	15	8	Z	Datum des Aufrufs [1]
4	17	23	6	Z	Uhrzeit des Aufrufs [2]

[1]Datum in der Form `yyyymmdd`

[2]Uhrzeit in der Form `hhmmss`

Länge der DSSM-Kennzeichnung: 29 Byte

3.6.1 ESMC - Subsystem-Initialisierungs-Abrechnungssatz

Der Abrechnungssatz wird bei jedem Ablauf der Initialisierungsphase eines Subsystems geschrieben. Diese Aktivierungsroutine durchläuft das Subsystem, unter Steuerung von DSSM, bei den Kommandos START-SUBSYSTEM und RESUME-SUBSYSTEM.

Maximale Länge des Subsystem-Initialisierungs-Abrechnungssatzes: 54 Byte

(A) Satzbeschreibung: Satzkennung: „ESMC“

(B) Kennzeichnungsteil: Subsystem-Kennzeichnung

(C) Grundinformation

Feld-Nr.	Distanz		Länge	Format	Bedeutung
	hex	dez	(Byte)		
1	00	0	1	B	Zustandsanzeige [1]
2	01	1	1	A	Kennzeichen Jahreszeit (aktuell) [2]
3	02	2	1	-	- reserviert -

[1] Die Zustandsanzeige kann zwei mögliche Werte annehmen:
1 Subsystem wird nach Wartezustand wieder gestartet (Kommando RESUME-SUBSYSTEM)
0 Subsystem ist gestartet (Kommando START-SUBSYSTEM)

[2] „S“ für Sommerzeit; „W“ für Winterzeit

Länge der Grundinformation: 3 Byte

(D) Variable Information

Die variable Information des Subsystem-Initialisierungs-Abrechnungssatzes enthält **keine** Satzerweiterung.

3.6.2 ESMD - Subsystem-Beendigungs-Abrechnungssatz

Der Abrechnungssatz wird bei jedem Ablauf der Deinitialisierungsphase eines Subsystems geschrieben. Diese Routine zur Beendigung durchläuft das Subsystem, unter Steuerung von DSSM, bei den Kommandos STOP-SUBSYSTEM und HOLD-SUBSYSTEM.

Maximale Länge des Subsystem-Beendigungs-Abrechnungssatzes: 54 Byte

(A) Satzbeschreibung: Satzkennung: „ESMD“

(B) Kennzeichnungsteil: Subsystem-Kennzeichnung

(C) Grundinformation

Feld-Nr.	Distanz		Länge	Format	Bedeutung
	hex	dez	(Byte)		
1	00	0	1	B	Zustandsanzeige [1]
2	01	1	1	A	Kennzeichen Jahreszeit (aktuell) [2]
3	02	2	1	-	- reserviert -

[1] Die Zustandsanzeige kann zwei mögliche Werte annehmen:
1 Subsystem wird in den Wartezustand versetzt (Kommando HOLD-SUBSYSTEM)
0 Subsystem ist beendet (Kommando STOP-SUBSYSTEM)

[2] „S“ für Sommerzeit; „W“ für Winterzeit

Länge der Grundinformation: 3 Byte

(D) Variable Information

Die variable Information des Subsystem-Beendigungs-Abrechnungssatzes enthält **keine** Satzerweiterung.

3.7 Fehlerbehandlung in DSSM

DSSM-Task-Error

1. DSSM beendet sich abnormal während des ersten Schrittes von Startup, z.B. wenn Subsysteme aktiviert werden, die mit dem Attribut *BEFORE-DSSM-LOAD oder *AT-DSSM-LOAD definiert wurden. Es wird ein Fehlercode zur Startup-Task geschickt. Normalerweise wird der Startup abgebrochen; u.U. kann er jedoch ohne DSSM-Initialisierung fortgeführt werden (abhängig von der Startup-Realisierung).

   Zu den Startup-Schritten siehe "MODIFY-SUBSYSTEM-ATTRIBUTES" oder Handbuch „Einführung in die Systembetreuung“.
2. DSSM beendet sich abnormal während des zweiten Schrittes von Startup, z.B. wenn Subsysteme aktiviert werden, die mit dem Attribut *MANDATORY-AT-STARTUP oder *BEFORE-SYSTEM-READY definiert wurden oder während die Datenstrukturen eines Subsystems mit dem Attribut *AFTER-SYSTEM-READY aktualisiert werden. Hat DSSM versucht, eines dieser Subsysteme zu aktivieren, wird das Subsystem in den Zustand LOCKED überführt.

   Wenn ein Subsystem mit *MANDATORY-AT-STARTUP in den Zustand LOCKED überführt wurde, wird ein Fehlercode zur Startup-Task geschickt. Ob der Startup abgebrochen wird, ist abhängig von der Startup-Realisierung.
3. DSSM beendet sich abnormal während Shutdown, wenn das zu deaktivierende Subsystem im Zustand LOCKED ist.
   Die Behandlung anderer Subsysteme wird von Shutdown normal fortgesetzt.
4. andere Ursachen, die zur abnormalen Beendigung von DSSM führen: DSSM analysiert die Situation, bereinigt seine internen Tabellen und trifft folgende Entscheidungen abhängig davon, wo der Fehler auftrat:

Der Fehler tritt auf während ...	**Reaktion**
CREATE/DELETE/RESUME/HOLD	Subsystem LOCKED (mit Fehlermeldung)
Austausch von Subsystemversionen	die fehlerhafte Routine wird nochmals aufgerufen; führt dies wiederum zum Fehler,
• mit Unterbrechung der Verfügbarkeit	• wird die betreffende Version in den Zustand LOCKED gesetzt,
• ohne Unterbrechung der Verfügbarkeit	• werden beide Subsystemversionen in den Zustand LOCKED gesetzt.
ADD-/MODIFY-SUBSYSTEM-PARAMETER, REMOVE-SUBSYSTEM	die Anweisung wird abgebrochen, wobei Teile des Katalogs bereits geändert sein können
Wiederherstellung der Holdertask	die Wiederherstellung wird abgebrochen
SHOW-SUBSYSTEM-INFO, SAVE-CATALOG	Anforderung wird abnormal beendet

   Nach dem Bereinigen der Tabellen setzt DSSM seine Arbeit mit den Anforderungen fort, die in der DSSM-Börse warten.

Holdertask-Error

Treten in der Holdertask Probleme auf oder wird sie abnormal beendet, analysiert DSSM die Situation, initiiert automatisch den Wiederanlauf der Holdertask und trifft folgende Entscheidungen:

Das Problem tritt auf ...	**Reaktion auf den Holdertask-Error bei**	
	RESTART-REQUIRED=*YES	**RESTART-REQUIRED=*NO**
- während normaler Anforderungen an das Subsystem		
Aktivierung	Subsystem LOCKED	Subsystem LOCKED
INIT-Routine	Aufruf der INIT-Routine	Subsystem LOCKED
CLOSE-CTRL-Routine	Subsystem CREATED	Subsystem LOCKED
STOPCOM-Routine	Aufruf der INIT-Routine	Subsystem LOCKED
DEINIT-Routine	Fortsetzung der Deinitialisierung	Subsystem LOCKED
Deaktivierung	Subsystem LOCKED	Subsystem LOCKED
Subsystem-Session (Arbeitstask)	Aufruf der INIT-Routine	Subsystem LOCKED
- beim Austausch von Subsystemversionen mit Unterbrechung der Verfügbarkeit		
Aktivierung V2	Subsystem V2 LOCKED und Abbruch des Austauschs	Subsystem V2 LOCKED und Abbruch des Austauschs
STOPCOM-Routine V1	Subsystem V1 LOCKED und Aufruf der INIT-Routine für V2	Subsystem V1 LOCKED und Aufruf der INIT-Routine für V2
INIT-Routine V2	Aufruf INIT-Routine V2 und Fortsetzung der DEINIT-Routine von V1	Subsystem V2 LOCKED und Fortsetzung der DEINIT-Routine von V1
Deaktivierung V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
DEINIT-Routine V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
- beim Austausch von Subsystemversionen ohne Unterbrechung der Verfügbarkeit		
Aktivierung V2	Subsystem V2 LOCKED und Abbruch des Austauschs	Subsystem V2 LOCKED und Abbruch des Austauschs
CLOSE-CTRL-Routine V1	Abbruch des Austauschs (Subsystem V1 CREATED)	Subsystem V1 LOCKED und Entladen von V1
INIT-Routine V2	Aufruf INIT-Routine V2 und Fortsetzung der DEINIT-Routine von V1	Subsystem V2 LOCKED und Abbruch des Austauschs

STOPCOM-Routine V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
DEINIT-Routine V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED
Deaktivierung V1	Subsystem V1 LOCKED	Subsystem V1 LOCKED

Fehlerdatei SERSLOG

DSSM schreibt einen Eintrag in die Error-Logging-Datei SERSLOG, wenn:

- ein Systemaufruf fehlerhaft ist
- inkonsistente Daten im internen Subsystemkatalog erkannt werden
- die DSSM-Task abnormal beendet wird (in diesem Fall enthält der SERSLOG-Eintrag eine Mitteilung, die die aktuelle Situation beschreibt)

Der Eintrag kann aus bis zu drei Teilen bestehen:

1. der Parameterliste des fehlerhaften Systemaufrufs oder den inkonsistenten Daten (in Einheiten zu 2 KByte)
2. dem Returncode (wenn er nicht schon Bestandteil der Parameterliste ist)
3. der Adresse der fehlerhaften Routine und die Adresse der Routine, die die fehlerhafte Routine aufgerufen hat. Dieser Eintrag ist für die Diagnose nützlich, denn DSSM benutzt einen eigenen zentralen SERSLOG-Aufruf.

3.8 Die Kommandos von DSSM

Kommando	Bedeutung
ADD-SUBSYSTEM	Dynamischen Subsystemkatalog erweitern
HOLD-SUBSYSTEM	Subsystem in den Wartezustand versetzen
MODIFY-SUBSYSTEM-PARAMETER	Subsystem-Parameter verändern
RELEASE-SUBSYSTEM-SPACE	Reservierten Adressraum von Subsystemen freigeben
REMOVE-SUBSYSTEM	Inaktives Subsystem aus dynamischem Katalog entfernen
RESUME-SUBSYSTEM	Wartezustand für ein Subsystem aufheben
SAVE-SUBSYSTEM-CATALOG	Änderungen des dynamischen Subsystemkatalogs sichern
SET-DSSM-OPTIONS	Protokollierungsfunktion von DSSM aus- oder einschalten
SHOW-DSSM-INFORMATION	Informationen über DSSM anzeigen
SHOW-RESTART-OPTIONS	Informationen über den automatischen Restart ausgeben
SHOW-SUBSYSTEM-ATTRIBUTES	Informationen über Subsystem-Attribute anzeigen
SHOW-SUBSYSTEM-INFO	Informationen über aktuelle Subsystem-Konfiguration anfordern
SHOW-SUBSYSTEM-STATUS	Informationen über Zustand von Subsystemen anfordern
START-SUBSYSTEM	Subsystem aktivieren
STOP-SUBSYSTEM	Subsystem deaktivieren
UNLOCK-SUBSYSTEM	Subsystem im Zustand LOCKED in den Zustand NOT-CREATED überführen

Tabelle 7: Kommandos von DSSM

Die Kommandos sind im Handbuch „BS2000 OSD/BC“ beschrieben. Dort finden Sie auch die **SDF-Syntaxdarstellung** der Kommandos.

3.9 Beispiel für Ausgabe in eine S-Variable

Das Beispiel zeigt die Ausgabe in S-Variablen in Abhängigkeit von der Privilegierung des Aufrufers. Es stellt keine reale Situation dar, sondern ist ein konstruiertes, vereinfachtes Beispiel zur Veranschaulichung der Problematik.

Folgende Privilegierungen sind gegeben:

- Fall 1: der Aufrufer hat nur das Privileg STD-PROCESSING
- Fall 2: der Aufrufer hat die Privilegien STD-PROCESSING sowie SUBSYSTEM-MANAGEMENT oder OPERATING
- Fall 3: der Aufrufer hat nicht das Privileg STD-PROCESSING, aber mindestens eines der Privilegien SUBSYSTEM-MANAGEMENT oder OPERATING

Folgende Konfiguration soll gegeben sein:

- das globale, nicht-privilegierte Subsystem AA V04.6 ist im Zustand CREATED; es ist eine Task angeschlossen (TID=00010054, TSN=0123)
- das globale, privilegierte Subsystem XX V01.0 ist im Zustand CREATED: es ist eine Task angeschlossen (TID=0002006F, TSN=0BFC)
- das globale, privilegierte Subsystem XX V02.0 ist im Zustand IN-DELETE: es sind zwei Tasks angeschlossen (TID=00070015, TSN=0CMM und TID=00010057, TSN=00AP)

Es wird eine zusammengesetzte Variable vom Typ Liste mit dem Namen `DATA` deklariert, und dem S-Variablenstrom SYSINF zugewiesen.

```
/DECLARE-VARIABLE VAR-NAME=DATA(TYPE=*STRUCTURE),
MULTIPLE-ELEMENTS=*LIST

/ASSIGN-STREAM STREAM-NAME=SYSINF,TO=*VAT(VAR-NAME=DATA)

/SHOW-STREAM-ASSIGNMENT

       STREAM-NAME   = SYSINF
        ASSIGN-LEVEL = 0

        DESTINATION  = *VARIABLE

          VARIABLE-NAME = DATA
               VAR-MODE = *EXTEND

          RETURN-VARIABLE-NAME = *NONE

          CONTROL-VAR-NAME = *NONE
          RET-CONTROL-VAR-NAME = *NONE

    STREAM-NAME   = SYSMSG
```

Fall 1:
der Aufrufer hat nur das Privileg STD-PROCESSING

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL
————————————————————————  (1)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED

% SUBSYSTEM AA     /V04.6    IS CREATED

/SHOW-VAR DATA
————————————————————————————————————————————————————————————  (2)
   :

   DATA(*LIST).SUBSYS-TYPE='*LOC'

   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'

   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.6'

   DATA(*LIST).SUBSYS-STA='*CREATED'

/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=AA
————————————————————————  (3)
% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED

% SUBSYSTEM AA      /V04.6    IS CREATED

/SHOW-VAR DATA
————————————————————————————————————————————————————————————  (4)

   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'

   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.5'
   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.6'

   DATA(*LIST).SUBSYS-STA='*CREATED'
```

(1) Es werden der Name, die Version und der Status aller globalen, nicht-privilegierten sowie aller lokalen Subsysteme nach SYSOUT ausgegeben. Im konstruierten Beispiel ist nur das globale Subsystem AA nicht-privilegiert. Es wird - zusammen mit dem lokalen Subsystem AA - ausgegeben.

(2) Die Ausgabe in der S-Variablen DATA enthält explizit den Subsystem-Typ „global“ oder „lokal“.

(3) Die Beschränkung in der Ausgabe auf das Subsystem AA ergibt keine geänderte Ausgabe nach SYSOUT im Vergleich zu (1).

(4) Die Beschränkung in der Ausgabe auf das Subsystem AA ergibt keine geänderte Ausgabe in die S-Variable DATA im Vergleich zu (2).

Fall 2:
der Aufrufer hat die Privilegien **STD-PROCESSING** sowie **SUBSYSTEM-MANAGEMENT** und/oder **OPERATING**

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL
————————————————————— (5)

% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED

% SUBSYSTEM AA     /V04.6    IS CREATED
% SUBSYSTEM XX     /V01.0    IS CREATED

% SUBSYSTEM XX     /V02.0    IS IN DELETE / WAIT-DISCON

/SHOW-VAR DATA
————————————————————————————————————————————————————— (6)
   :
   DATA(*LIST).SUBSYS-TYPE='*LOC'

   DATA(*LIST).SUBSYS-NAME='AA'
   DATA(*LIST).SUBSYS-VERSION='04.5'

   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.6'

   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='XX'

   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='XX'

   DATA(*LIST).SUBSYS-VERSION='02.0'
   DATA(*LIST).SUBSYS-STA='*IN-DELETE'

   DATA(*LIST).SUBSYS-INT-STA='WAIT-DISCON'

/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=AA
————————————————————————— (7)

% LOCAL SUBSYSTEM AA     /V04.5    IS CREATED
```

```
% SUBSYSTEM AA     /V04.6    IS USED BY 1 TASK
% TASKID 00010054

% TSN        0123

%   4 CONNECTIONS SINCE STARTUP
/SHOW-VAR DATA
_____________________________________________________________ (8)

   :

   DATA(*LIST).SUBSYS-TYPE='*LOC'
   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.5'

   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.6'
   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).CONN-NUM-SINCE-START=4

   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00010054'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='0123'
```

(5) Es werden der Name, die Version und der Status aller globalen sowie aller lokalen Subsysteme nach SYSOUT ausgegeben. Es wird also über alle Subsysteme informiert, die im laufenden System geladen sind. Im konstruierten Beispiel sind das die globalen Subsysteme XX (privilegiert, in zwei Versionen) und AA (nicht-privilegiert) sowie das lokale Subsystem AA.

(6) Die Ausgabe in der S-Variablen DATA enthält explizit den Subsystem-Typ „global“ oder „lokal“.

(7) Die Beschränkung in der Ausgabe auf das Subsystem AA ergibt folgende Änderungen im Vergleich zu (1): Sind an ein Subsystem Tasks angeschlossen, werden sie mit ihrer TSN und TID angezeigt. Die Anzahl aller Tasks, die seit Startup an dieses Subsystem konnektiert waren, wird ebenfalls nach SYSOUT ausgegeben.

(8) Die Ausgabe der Subsysteme AA in die S-Variable DATA enthält die gleichen Informationen wie in der SYSOUT-Ausgabe, ergänzt durch den Subsystem-Typ „global“ oder „lokal“.

Fall 3:
der Aufrufer hat nicht das Privileg STD-PROCESSING, aber mindestens eines der Privilegien **SUBSYSTEM-MANAGEMENT** oder **OPERATING**

```
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=*ALL
————————————————————— (9)
% SUBSYSTEM AA     /V04.6    IS CREATED

% SUBSYSTEM XX     /V01.0    IS CREATED

% SUBSYSTEM XX     /V02.0    IS IN DELETE / WAIT-CLS-CTRL
/SHOW-VAR DATA ————————————————————————————————————————————
(10)

   :

   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='AA'

   DATA(*LIST).SUBSYS-VERSION='04.6'

   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='XX'

   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'

   DATA(*LIST).SUBSYS-TYPE='*GLB'

   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='02.0'

   DATA(*LIST).SUBSYS-STA='*IN-DELETE'

   DATA(*LIST).SUBSYS-INT-STA='WAIT-CLS-CTRL'
/SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=XX ——————————————————————
(11)

% SUBSYSTEM XX     /V0.0     IS USED BY 1 TASK

% TASKID 0002006F

% TSN        0BFC

%   7 CONNECTIONS SINCE STARTUP

% SUBSYSTEM XX     /V02.0    IS USED BY 2 TASKS
% TASKID 00070015 00010057
```

```
% TSN        0CMM     0OAP
%                         IS IN DELETE / WAIT-DISCON
% 12 CONNECTIONS SINCE STARTUP
```

/SHOW-VAR DATA ——————————————————————————————————————

(12)

```
   :
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='01.0'
   DATA(*LIST).SUBSYS-STA='*CREATED'
   DATA(*LIST).CONN-NUM-SINCE-START=7
   DATA(*LIST).USED-TASK-LIST(1).TID='0002006F'
   DATA(*LIST).USED-TASK-LIST(1).TSN='0BFC'
   DATA(*LIST).SUBSYS-TYPE='*GLB'
   DATA(*LIST).SUBSYS-NAME='XX'
   DATA(*LIST).SUBSYS-VERSION='02.0'
   DATA(*LIST).SUBSYS-STA='*IN-DELETE'
   DATA(*LIST).SUBSYS-INT-STA='WAIT-DISCON'
   DATA(*LIST).CONN-NUM-SINCE-START=12
   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00070015'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='0CMM'
   DATA(*LIST).USED-TASK-LIST(*LIST).TID='00010057'
   DATA(*LIST).USED-TASK-LIST(*LIST).TSN='00AP'
```

(9) Es werden der Name, die Version und der Status aller globalen Subsysteme nach SYSOUT ausgegeben. Im konstruierten Beispiel sind das die globalen Subsysteme XX (privilegiert, in zwei Versionen) und AA (nicht-privilegiert).

(10) Die Ausgabe in die S-Variable DATA beinhaltet die gleichen Informationen wie nach SYSOUT.

(11) Die Beschränkung in der Ausgabe auf das Subsystem XX ergibt folgende Änderungen im Vergleich zu (9): Sind an ein Subsystem Tasks angeschlossen, werden sie mit ihrer TSN und TID angezeigt. Die Anzahl aller Tasks, die seit Startup an dieses Subsystem konnektiert waren, wird ebenfalls nach SYSOUT ausgegeben.

(12) Die Ausgabe der Subsysteme XX in die S-Variable DATA enthält die gleichen Informationen wie in der SYSOUT-Ausgabe, ergänzt durch den Subsystem-Typ „global“.

4 SSCM

Das Subsystem SSCM (Static Subsystem Catalog Manager) ermöglicht eine flexible und anwenderfreundliche Verwaltung des statischen Subsystemkatalogs (SSMCAT).

Zur Installation von SSCM und zur Koexistenz der SSCM-Versionen, siehe "Installation von SSCM".

4.1 Generierung eines Subsystemkatalogs

Der zu erzeugende Subsystemkatalog muss auf das Home-Pubset gebracht und unter der Benutzerkennung TSOS abgelegt werden. Der Name des Katalogs ist frei wählbar und kann über den Parameterservice (siehe "Inbetriebnahme der dynamischen Subsystemverwaltung") dem System bekannt gemacht werden. Für die Generierung eines Subsystemkatalogs steht das Programm SSCM zur Verfügung.
Auf folgendem Bild ist ein funktionaler Ablauf der Generierung eines Subsystemkatalogs skizziert:

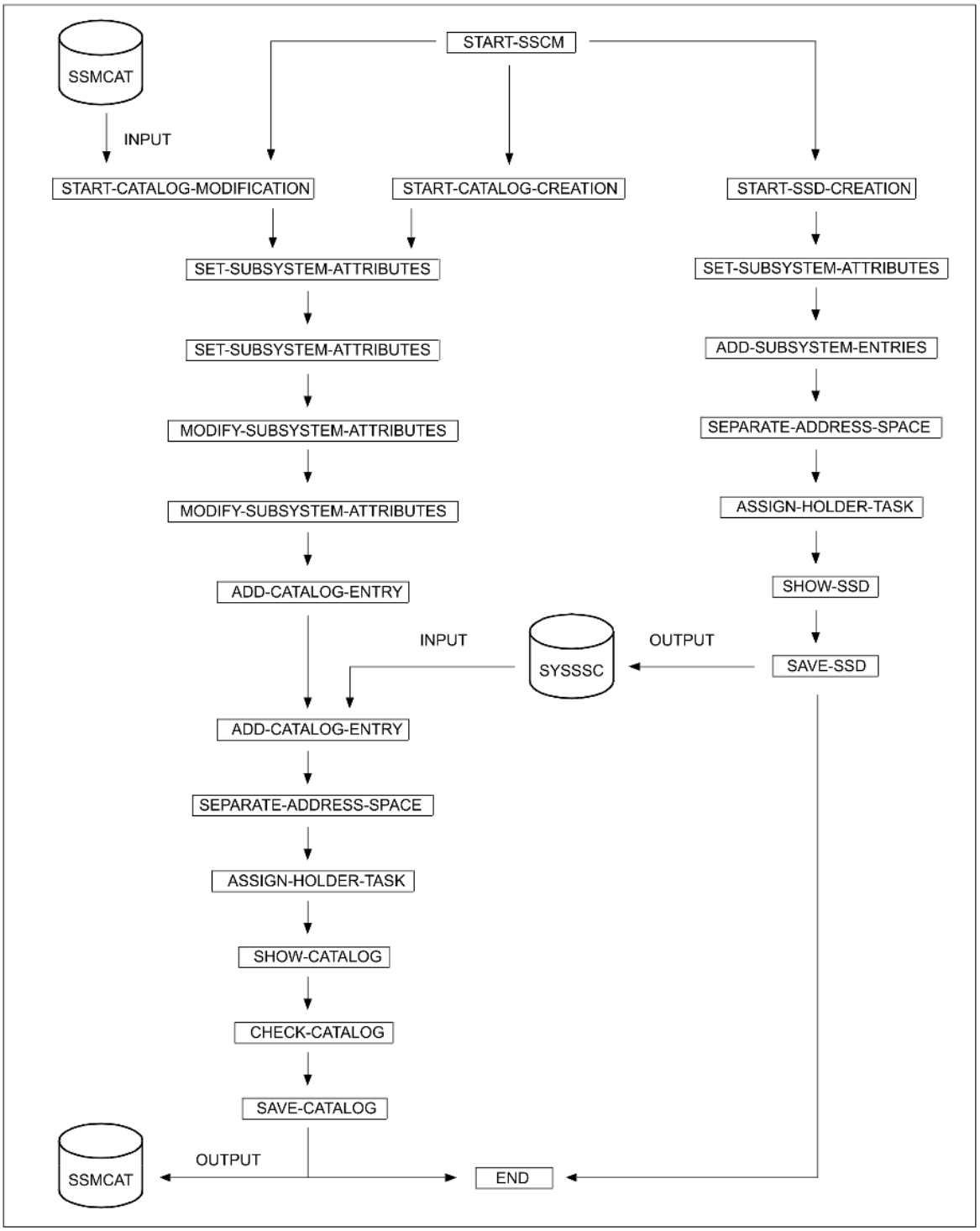

Bild 1: Ablauf einer Subsystemkatalog-Generierung

4.2 Starten und Beenden von SSCM

Das Programm SSCM wird mit folgendem Kommando gestartet.

START-**SSC**M
MONJV = ***NONE** / <filename 1..54 without-gen-vers> ,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767>

SSCM kann ebenso mit der Abkürzung **SSCM** aufgerufen werden.

Die Meldungsdatei für SSCM ($TSOS.SYSMES.SSCM.210) wird aktiviert und das in der Modulbibliothek SYSLNK.SSCM.210 stehende Bindelademodul SSCM wird geladen (Meldung `BLS0517`).

Die Anweisung **END** beendet SSCM.

Programmüberwachung durch Monitor-Jobvariable

Die programmüberwachende Jobvariable muss im Kommando START-SSCM mit MONJV=<jv-name> vereinbart werden.

Sie kann von SSCM mit folgenden Werten versorgt werden:

`'$T0000'`	Programm wurde normal beendet.
`'$T1010'`	Anweisung abgewiesen, das Programm wird fortgesetzt.
`'$A2010'`	Anweisung abgewiesen, das Programm wurde beendet.
`'$A2015'`	Unerwartetes EOF auf SYSDTA, das Programm wurde abgebrochen.
`'$A3020'`	SSCM-interner Fehler, das Programm wird abgebrochen.

Zur Programmüberwachung durch Monitor-Jobvariablen siehe auch Handbuch „B2000 OSD/BC“ und „Jobvariablen“.

Überwachung durch Auftragsschalter

Wird eine Anweisung von SSCM abnormal beendigt, wird der Auftragsschalter 31 eingeschaltet.

Ist der Auftragsschalter 1 eingeschaltet, führt SSCM Anweisungen nicht durch, es findet nur eine Syntaxprüfung statt.

4.3 Die Anweisungen von SSCM

SSCM wird mit einer SDF-Schnittstelle für die Benutzeroberfläche angeboten.
Dem Anwender stehen somit alle Funktionen und Vorzüge, die SDF bietet - geführter Dialog, Hilfe-Texte für Operanden, Einsatz von Default-Werten - auch für die Nutzung von SSCM zur Verfügung.

Die **SDF-Syntaxdarstellung** der Kommandos ist im Handbuch „BS2000 OSD/BC“ erläutert.

Folgende SSCM-Anweisungen stehen zur Verfügung:

Anweisung	Bedeutung
ADD-CATALOG-ENTRY	Subsystemdefinition(en) in den Katalog aufnehmen
ADD-SUBSYSTEM-ENTRIES	Zusätzliche Auftragseingänge definieren
ASSIGN-HOLDER-TASK	Subsysteme auf Holdertasks verteilen
CHECK-CATALOG	Subsystemdefinition(en) auf Konsistenz prüfen
GENERATE-CATALOG-SOURCE	SSCM-Anweisungsliste für Generierung erstellen
MODIFY-SUBSYSTEM-ATTRIBUTES	Eigenschaften von Subsystemen verändern
MODIFY-WORK-TASK-ATTRIBUTE	Arbeitstask-Parameter verändern
REMOVE-ADDR-SPACE-SEPARATION	Disjunkte Verteilung der Subsysteme im Klasse-5-Speicher aufheben
REMOVE-CATALOG-ENTRY	Subsystemdefinition logisch aus dem Subsystemkatalog löschen
SAVE-CATALOG	Subsystemkatalog als PAM-Datei abspeichern
SAVE-SSD	Subsystemdefinition(en) abschließen
SEPARATE-ADDRESS-SPACE	Disjunkte Verteilung der Subsysteme im Klasse-5-Speicher steuern
SET-SUBSYSTEM-ATTRIBUTES	Eigenschaften und Einsprungstellen eines Subsystems definieren
SHOW-CATALOG	Subsystem-Konfiguration anzeigen
SHOW-SSD	Inhalt eines SSD-Objects (Subsystemdefinitionen) anzeigen
START-CATALOG-CREATION	Name eines statischen Subsystemkatalogs vereinbaren
START-CATALOG-MODIFICATION	Statischen Subsystemkatalog verändern
START-SSD-CREATION	SSD-Object zur Aufnahme von Subsystemdefinitionen generieren

Tabelle 8: SSCM-Anweisungen

4.3.1 ADD-CATALOG-ENTRY

Subsystemdefinition(en) in den Subsystemkatalog aufnehmen

Funktionsbeschreibung

Mit dieser Anweisung werden die Definitionen neuer Subsysteme, die in einem SSD-Object hinterlegt sind, in den eröffneten Subsystemkatalog aufgenommen.

Die Anweisung wird zurückgewiesen, wenn kein aktueller Katalog, in den die Definitionen integriert werden sollen, mit der Anweisung START-CATALOG-CREATION oder START-CATALOG-MODIFICATION eröffnet wurde. Die Definition für ein Subsystem, das im eröffneten Katalog bereits enthalten ist, wird ignoriert; die Verarbeitung wird allerdings nach Ausgabe einer entsprechenden Meldung fortgesetzt.

Format

ADD-CATALOG**-ENTRY**
FROM-FILE = <filename 1..54 without-gen-vers> ,**INSTAL**LATION**-USERID** = ***UNCHA** NGED / ***DEF**AULT**-USERID** / <name 1..8> ,**CORR**ECTION**-STATE** = ***UNCHA** NGED / <c-string 3..3> / <text 3..3>

Operandenbeschreibung

FROM-FILE = <filename 1..54>
Name der Datei, in der die Subsystemdefinition(en) gesucht werden. Die Datei muss ein von SSCM generiertes SSD-Object vom Typ ISAM sein, in der die Eigenschaften eines oder mehrerer Subsysteme hinterlegt sind.

INSTALLATION-USERID =
Vereinbart eine Benutzerkennung, unter der die Nebenkomponenten des Subsystems (Rep-Datei, Objektmodulbibliothek, Meldungsdatei, Syntaxdatei und Subsystem-Informationsdatei) erwartet werden, falls diese Dateien noch keiner Benutzerkennung zugeordnet sind.

INSTALLATION-USERID = *UNCHANGED
Voreinstellung: die Dateien werden unter der Benutzerkennung erwartet, die bei der Subsystemdefinition (Anweisung SET-SUBSYSTEM-ATTRIBUTES) angegeben wurde.

INSTALLATION-USERID = *DEFAULT-USERID
Die Dateien werden unter der System-Standardkennung erwartet (Prefix „$.“).

INSTALLATION-USERID = <name 1..8>
Benutzerkennung, unter der die Dateien erwartet werden. Eine von dieser Angabe abweichende Vereinbarung, die in der Anweisung SET-SUBSYSTEM-ATTRIBUTES für ein SSD-Object getroffen wurde, wird überschrieben.

CORRECTION-STATE =
Ersetzt im Katalog die letzten drei Zeichen der Subsystemversion aus der SSD-Datei, die den Freigabe- und Korrekturstand darstellen.
Ist die Subsystemversion in der SSD-Datei mit vier Zeichen angegeben, werden die bei CORRECTION-STATE angegebenen drei Zeichen mit diesen verkettet.

CORRECTION-STATE = *UNCHANGED
Voreinstellung: Freigabe- und Korrekturstand bleiben unverändert.

CORRECTION-STATE = <text 3..3>
Angabe des Freigabe- und Korrekturstandes im Format: aso . Dabei bedeutet:

a: Freigabestand; alphabetisches Zeichen
so: Korrekturstand; numerische Zeichen

CORRECTION-STATE = <c-string 3..3>
Angabe des Freigabe- und Korrekturstandes als Zeichenkette im Format: aso
Zur Bedeutung siehe CORRECTION-STATE = <text 3..3> .

Hinweise

- Enthält eine Subsystemdefinition Dateinamen ohne Benutzerkennung und ist keine Installationskennung angegeben, sucht die zuständige DSSM-Task die Dateien unter der Benutzerkennung TSOS bzw. unter der Benutzerkennung der aufrufenden Task im Fall eines lokalen Subsystems.
  Sind die Dateien unter einer anderen Kennung abgelegt, so muss diese beim Operanden INSTALLATION-USERID in einem der Kommandos ADD-CATALOG-ENTRY, SET-SUBSYSTEM-ATTRIBUTES oder MODIFY-SUBSYSTEM-ATTRIBUTES angegeben werden.
- Wird bei FROM-FILE ein SSD-Object angegeben, das mehrere Subsystemdefinitionen enthält, werden die Freigabe- und Korrekturstände aller in diesem SSD-Object definierten Subsysteme durch die Werte der Operanden CORRECTION-STATE und INSTALLATION-USERID beeinflusst.
- Die Dateinamen eines Subsystems werden nicht geändert, wenn Freigabe- und Korrekturstand angegeben sind. Wünscht der Anwender eine Namensänderung, muss er dies mit der Anweisung MODIFY-SUBSYSTEM-ATTRIBUTES veranlassen.
- Wird der Operand CORRECTION-STATE angegeben, kann es bei der Versionsprüfung für Subsysteme, die voneinander abhängig sind, oder für Subsysteme, die untereinander Adressbeziehungen erhalten, zu einem CHECK-CATALOG-Fehler kommen. Das kann durch geeignete Angaben bei der Anweisung MODIFY-SUBSYSTEM-ATTRIBUTES MODIFY-REFER-SUBS=..., MODIFY-RELATED-SUBS=... vermieden werden.

4.3.2 ADD-SUBSYSTEM-ENTRIES

Zusätzliche Auftragseingänge definieren

Funktionsbeschreibung

Mit dieser Anweisung ist es möglich, für ein Subsystem in einem SSD-Object mehr als die bereits bei SET-SUBSYSTEM-ATTRIBUTES angegebenen und maximal erlaubten 100 Auftragseingänge zu definieren.

Mit jeder Anweisung ADD-SUBSYSTEM-ENTRIES (sie kann für ein- und dasselbe Subsystem mehrmals ausgeführt werden) können bis zu 100 weitere Auftragseingänge für ein einzelnes Subsystem definiert werden.

ADD-SUBSYSTEM-ENTRIES wird zurückgewiesen, wenn zuvor die Anweisung START-SSD-CREATION nicht ausgeführt wurde.

Die Anweisung wird abgewiesen, wenn das angegebene Subsystem bei der Anweisung SET-SUBSYSTEM-ATTRIBUTES oder der Anweisung MODIFY-SUBSYSTEM-ATTRIBUTES mit dynnamisch zu versorgenden Einsprungstellen (*BY-PROGRAM(...)) definiert wurde.

Die Ausführung der Anweisung wird abgebrochen, wenn

- das mit TO-SUBSYSTEM und VERSION bestimmte Subsystem nicht im aktuellen SSD-Object gefunden wird
- ein bereits existierender Auftragseingang ein zweites Mal definiert werden soll

Es wird eine Fehlermeldung ausgegeben.

Format

ADD-SUBSYSTEM-ENTRIES

TO-SUBSYSTEM = <structured-name 1..8>(...)

<structured-name 1..8>(...)

VERSION = <c-string 3..8> / <text 3..8> “

,**SUBSYSTEM-ENTRIES** = list-poss(100): <text 1..8>(...)

<text 1..8>(...)

MODE = ***LINK** / ***ISL**(...) / ***SVC**(...) / ***SYSTEM-EXIT**(...)

***ISL**(...)

FUNCTION-NUMBER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SVC**(...)

NUMBER = <integer 0..255>

,**CALL-BY-SYSTEM-EXIT** = ***ALLOW** ED / ***FORBIDDEN**

,**FUNCTION-NUMBER** = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SYSTEM-EXIT**(...)

NUMBER = <integer 0..127>

,**CONNECTION-ACCESS** = ***ALL** / ***SYS**TEM / ***SIH**

,**CONNECTION-SCOPE** = ***TASK** / ***PROG**RAM / ***FREE** / ***CALL** / ***OPTIMAL**

, **FIRST-CONNECTION** = ***ALLOW** ED / ***FORBIDDEN**

Operandenbeschreibung

TO-SUBSYSTEM = <structured-name 1..8>(...)
Vereinbart Name und Version des Subsystems, für das weitere Auftragseingänge definiert werden sollen.

VERSION = <c-string 3..8> / <text 3..8>
Die Version des Subsystems ist im Format „[V][m]m.n[aso]“ zu vereinbaren, wobei die Textteile folgende Bedeutung besitzen:

mm: Hauptversion (numerisch)

n: Nachtragsversion (numerisch)

aso: Änderungsstand (a: Buchstabe, Freigabestand; so: numerisch, Korrekturstand)

SUBSYSTEM-ENTRIES =
Vereinbart weitere Einsprungstellen (Auftragseingänge), die mit dem Subsystem verbunden sind. Es können bis zu 100 Auftragseingänge pro Anweisung vereinbart werden.

SUBSYSTEM-ENTRIES = list-poss(100): <text 1..8>
Vereinbart durch Angabe des Namens der Einsprungstelle maximal 100 Auftragseingänge, deren Typ jeweils in den Unterstrukturen definiert werden muss.

MODE =
Legt den Typ eines vereinbarten Auftragseingangs für das Subsystem fest.

MODE = *LINK
Voreinstellung: der Auftragseingang kann nicht über indirektes Linkage erreicht werden, sondern nur über eine CONNECT-Beziehung mittels eines externen Binder-Symbols.
Bei verschiedenen Versionen des gleichen Subsystems, die das gleiche externe Binder-Symbol nutzen, stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.

MODE = *ISL(...)
Der Auftragseingang wird durch indirekte Verbindung über System Procedure Linkage (nur für privilegierte Subsysteme) erreicht. Wird zusätzlich noch eine Funktions- und Versionsnummer der ISL-Einsprungstelle spezifiziert, darf sich die Kombination aus Name der Einsprungstelle, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED, siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES) decken.

Bei ungleichen Subsystemen, deren Auftragseingang über die selbe ISL-Einsprungstelle erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die dieselbe ISL-Einsprungstelle nutzen (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die dieselbe ISL-Einsprungstelle nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist, ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standard Header der Parameterliste des Aufrufers hinterlegt ist.
Die Angabe CONNECTION-ACCESS=*ALL (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES) ist für ISL-Einsprungstellen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der ISL-Einsprungstelle angesprochen werden soll, da die gleiche ISL-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Voreinstellung: es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der ISL-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten ISL-Funktionsnummer.

MODE = *SVC(...)
Der Auftragseingang wird durch indirekte Verbindung über Supervisor Call (SVC) erreicht.
Wird zusätzlich noch eine Funktions- und Versionsnummer der SVC-Einsprungstelle spezifiziert, darf sich die Kombination aus SVC-Nummer, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED, siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES) decken.

Bei ungleichen Subsystemen, deren Auftragseingang über den selben SVC erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist, ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standardheader der Parameterliste des Aufrufers hinterlegt ist.
Bei Angabe dieses Operandenwertes sollte der Operand CONNECTION-ACCESS vorzugsweise mit dem Wert *SYSTEM statt *ALL belegt werden (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES).

NUMBER = <integer 0..255>
Nummer des SVCs, über den der Auftragseingang erreicht wird. In Verbindung mit CONNECTION-ACCESS=*ALL (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES) ist die Verwendung einer SVC-Nummer größer 191 nicht zulässig.

CALL-BY-SYSTEM-EXIT =
Legt fest, ob die angegebene SVC-Nummer von System-Exit-Routinen aus aufgerufen werden darf.

CALL-BY-SYSTEM-EXIT = *ALLOWED
Voreinstellung: der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen zulässig.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
Der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der SVC-Einsprungstelle angesprochen werden soll, da die gleiche SVC-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Voreinstellung: es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der SVC-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten SVC-Funktionsnummer.

MODE = SYSTEM-EXIT(...)
Der Auftragseingang wird durch indirekte Verbindung über System-Exit-Routinen erreicht.
In Verbindung mit CONNECTION-ACCESS=*ALL (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES) ist die Verwendung dieses Operanden nicht zulässig.

NUMBER = <integer 0..127>
Nummer der System-Exit-Routine.

CONNECTION-ACCESS =
Vereinbart die Zugriffsberechtigung (Privilegierung) für das Subsystem.

CONNECTION-ACCESS = *ALL
Voreinstellung: privilegierte und nicht-privilegierte Programmläufe sind zugriffsberechtigt.
In Verbindung mit MODE=*SYSTEM-EXIT, MODE=*ISL und MODE=*SVC (mit einer SVC-Nummer größer 191) ist die Verwendung dieses Operandenwertes nicht zulässig (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES).

CONNECTION-ACCESS = *SYSTEM
Nur privilegierte Programmläufe sind zugriffsberechtigt.

CONNECTION-ACCESS = *SIH
Nur Tasks, die im Funktionszustand SIH laufen, sind zugriffsberechtigt.
Das aufgerufene Subsystem läuft ebenfalls im Funktionszustand SIH, d.h. es ist nicht unterbrechbar.
Die Angabe dieses Operandenwertes ist nur für Subsysteme zulässig, deren Auftragseingang definiert wird über

- System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
- CONNECTION-SCOPE=*OPTIMAL
- MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Bezeichnet das Ereignis, das die automatische Auflösung des Anschlusses an den angegebenen Subsystem-Auftragseingang hervorruft.

CONNECTION-SCOPE = *TASK
Voreinstellung: der Anschluss wird bei Taskbeendigung aufgehoben.

CONNECTION-SCOPE = *PROGRAM
Der Anschluss wird spätestens bei Programmbeendigung aufgehoben.
Zusammen mit MEMORY-CLASS=*LOCAL-UNPRIVILEGED darf nur CONNECTION-SCOPE=*PROGRAM angegeben werden (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES).

Die Angabe dieses Operandenwertes wird für Subsysteme empfohlen, die mit SUBSYSTEM-ACCESS=*LOW/*HIGH deklariert wurden.

CONNECTION-SCOPE = *FREE
DSSM soll keine Kontrolle von Anschlüssen zu diesen Auftragseingängen durchführen. Der Anschluss wird - außer bei einer expliziten Anforderung - nicht automatisch aufgelöst. Um Probleme oder mögliche Fehler beim Entladen des Subsystems zu vermeiden, müssen die Anschlüsse vom Subsystem selbst verwaltet werden.

CONNECTION-SCOPE = *CALL
Nach Rückkehr aus diesem Auftragseingang führt DSSM automatisch die Auflösung der Anschlüsse durch.

Dieser Operandenwert steht nur für Subsysteme zur Verfügung, deren Auftragseingang über System Procedure Linkage (ISL) oder Supervisor Call (SVC) definiert wird.

CONNECTION-SCOPE = *OPTIMAL
Das Subsystem wird deaktiviert bzw. angehalten, wenn keine Task mehr Anschluss zu diesem Auftragseingang hat.
Eine Routine, deren Einsprungstelle mit *OPTIMAL definiert wird, muss mit RETURN beendet werden.
Wird ein Auftragseingang eines Subsystems mit CONNECTION-SCOPE=*OPTIMAL definiert, sollten alle seine Auftragseingänge im Subsystemkatalog mit MODE=*LINK definiert werden.
Während ein Subsystem deaktiviert oder angehalten wird, wird kein Aufruf des Subsystems mit CONNECTION-SCOPE=*OPTIMAL akzeptiert. Ausnahme: das Subsystem wurde mit CREATION-TIME=*AT-SUBSYSTEM-CALL definiert und die aufrufende Task ist bereits mit ihm verbunden (siehe Anweisung SET-SUBSYSTEM-ATTRIBUTES).

FIRST-CONNECTION =
Bestimmt, ob der Erst-Anschluss der Task an den angegebenen Auftragseingang des Subsystems erlaubt ist.
Mindestens ein Auftragseingang eines Subsystems muss mit FIRST-CONNECTION = *ALLOWED definiert werden.

FIRST-CONNECTION = *ALLOWED

Voreinstellung: Der Erst-Anschluss an den angegebenen Auftragseingang ist erlaubt.

FIRST-CONNECTION = *FORBIDDEN

Der Anschluss an den angegebenen Auftragseingang über SVC oder ISL ist nicht erlaubt, wenn die Task nicht bereits an einen anderen Auftragseingang des Subsystems angeschlossen ist.
Die Angabe dieses Operandenwertes ist für Auftragseingänge, die mit MODE=*LINK/*SYSTEM-EXIT oder CONNECTION-ACCESS=*SIH definiert wurden, nicht erlaubt.

4.3.3 ASSIGN-HOLDER-TASK

Subsysteme auf Holdertasks verteilen

Funktionsbeschreibung

Mit dieser Anweisung wird die Verteilung der Subsysteme auf Holdertasks gesteuert. Die in der Anweisung aufgeführten Subsysteme können die Holdertask als Arbeitstask nutzen oder werden gemeinsam in einer Holdertask angelegt. Wird bei der Erstellung eines Katalogs auf diese Anweisung verzichtet, nimmt SSCM eine Standardaufteilung der Subsysteme vor, um die Zahl der Holdertasks zu begrenzen.

Für Subsysteme, die die Holdertask als Arbeitstask nutzen, ist die Anweisung verpflichtend.

Um Arbeitstask zu werden, müssen die Auftragseingänge des Subsystems mit einer der folgenden Kombinationen definiert sein:

CLOSE-CTRL	STOPCOM	DEINIT
*DYNAMIC	*DYNAMIC	*DYNAMIC
*DYNAMIC	*NO	*DYNAMIC
*NO	*DYNAMIC	*DYNAMIC *
*NO	*NO	*DYNAMIC *

*Aus Kompatibilitätsgründen ist es für ein Subsystem ohne CLOSE-CTRL-Routine nicht Pflicht, eine DEINIT-Routine zu definieren. Es ist jedoch darauf zu achten, dass in einem solchen Fall keine Garantie für den ordnungsgemäßen Ablauf des Subsystems gegeben werden kann.

Die Anweisung darf bei einem SSD-Object nur einmal pro Subsystem verwendet werden. Bei zwei aufeinander folgenden Anweisungen für das gleiche Subsystem hat im Konfliktfall - sich widersprechende Vereinbarungen - die Definition Vorrang, wonach die Holdertask als Arbeitstask genutzt werden soll.
Die Vereinbarung bezüglich einer gemeinsamen Holdertask gilt jeweils für alle Versionen des Subsystems mit Ausnahme der Versionen, die die Holdertask als Arbeitstask nutzen.

ASSIGN-HOLDER-TASK wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-SSD-CREATION
- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Format

ASSIGN-HOLDER-TASK

TYPE = **<u>*WORK-TASK</u>** (...) / ***SHARED-HOLDER**(...)

***WORK-TASK**(...)

SUBSYSTEM-NAME = <structured-name 1..8>

,**SUBSYS**TEM**-VERS**ION = <c-string 3..8> / <text 3..8>

,**TSN** = **<u>*BY-DSSM</u>** / <alphanum-name 1..4>

***SHARED-HOLDER**(...)

BY-SUBSYSTEMS = list-poss(15): <structured-name 1..8>

,**TSN** = **<u>*BY-DSSM</u>** / <alphanum-name 1..4>

Operandenbeschreibung

TYPE =
Legt fest, ob die Holdertask als Arbeitstask genutzt oder das Subsystem in einer gemeinsamen Holdertask angelegt werden soll.

TYPE = <u>*WORK-TASK</u>(...)
Voreinstellung: die Holdertask soll als Arbeitstask genutzt werden.

SUBSYSTEM-NAME = <structured-name 1..8>
Name des Subsystems, das die Holdertask als Arbeitstask nutzen soll.

SUBSYSTEM-VERSION = <c-string 3..8> / <text 3..8>
Version des Subsystems, das die Holdertask als Arbeitstask nutzen soll. Die Version muss bereits deklariert sein.

TSN =
Bestimmt, welche Task Sequence Number (TSN) die Arbeitstask des Subsystems erhalten soll.

TSN = <u>*BY-DSSM</u>
Voreinstellung: die TSN wird beim Laden der Arbeitstask von DSSM vergeben. Die TSN ist bekannt, da sie von DSSM berechnet wird. Sowohl die SYSLST-Ausgabe des Befehls SHOW-SUBSYSTEM-INFO als auch die Listenvariablenausgabe des Befehls SHOW-SUBSYSTEM-ATTRIBUTES zeigen den TSN-Wert an, unabhängig vom Erstellungsstatus des Subsystems.

TSN = <alphanum-name 1..4>
TSN, die die Arbeitstask beim Starten erhalten soll. Die angegebene TSN muss eindeutig festgelegt und verwendbar sein, wenn das Subsystem geladen wird.

TYPE = *SHARED-HOLDER(...)
Das Subsystem soll in einer gemeinsamen Holdertask angelegt werden.

BY-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
Name der maximal 15 Subsysteme, die in der gleichen Holdertask angelegt werden müssen.
Das erste in der Liste aufgeführte Subsystem muss bereits deklariert sein.

TSN =
Bestimmt, welche Task Sequence Number (TSN) die gemeinsame Holdertask erhalten soll.

TSN = *BY-DSSM

Voreinstellung: die TSN wird beim Laden der Arbeitstask von DSSM vergeben. Die TSN ist bekannt, da sie von DSSM berechnet wird. Befindet sich mindestens eines der Subsysteme in dieser gemeinsamen Holdertask im Status „Erstellt“, wurde die TSN bereits vergeben. Sowohl die SYSLST-Ausgabe des Befehls „SHOW-SUBSYSTEM-INFO“ als auch die Listenvariablenausgabe des Befehls „SHOW-SUBSYSTEM-ATTRIBUTES“ zeigen den TSN-Wert an, unabhängig vom Erstellungsstatus des Subsystems.

TSN = <alphanum-name 1..4>

TSN, die die gemeinsame Holdertask beim Starten erhalten soll. Die angegebene TSN muss eindeutig festgelegt und verwendbar sein, wenn das Subsystem geladen wird.

4.3.4 CHECK-CATALOG

Subsystemdefinition(en) auf Konsistenz prüfen

Funktionsbeschreibung

Mit dieser Anweisung werden die in einem Katalog zusammengefassten Subsystemdefinitionen auf Konsistenz geprüft.

CHECK-CATALOG wird zurückgewiesen, wenn der anzugebende Dateiname nicht existiert oder der Subsystemkatalog ohne Inhalt ist.

Wenn der angegebene Dateiname nicht dem des aktuell eröffneten Kataloges entspricht, wird folgende Meldung ausgegeben:

```
SCM0011    WOLLEN SIE WIRKLICH DEN KATALOG IM SPEICHER ÜBERSCHREIBEN ? (Y/N)
```

Bei Antwort **Y** gehen die Definitionen des aktuellen Katalogs verloren.
Bei Antwort **N** wird die Ausführung der Anweisung CHECK-CATALOG abgebrochen. Der Anwender kann mit SAVE-CATALOG die bisher nicht gespeicherten Subsystemdefinitionen in einer Datei ablegen.

> ¡ Ohne vorherige Prüfung der Link- und Abhängigkeitsbeziehungen kann der Katalog nicht gesichert werden.

Format

CHECK-CATALOG
CATALOG-NAME = ***CURR** ENT / <filename 1..54 without-gen-vers>
,**DEPENDENCE-RELATION** = ***Y** ES / ***NO**
,**LINK-RELATION** = ***Y** ES / ***NO**
,**RELATED-FILES** = ***NO** / ***Y**ES
,**OUTPUT** = ***SYSOUT** / ***SYSLST**(...)
***SYSLST**(...)
SYSLST-NUMBER = ***STD** / <integer 1..99>

Operandenbeschreibung

CATALOG-NAME =
Vereinbart den Namen des Kataloges, in dem die zu prüfenden Definitionen hinterlegt sind.

CATALOG-NAME = *CURRENT
Voreinstellung: der aktuell geöffnete Katalog (Anweisung START-CATALOG-CREATION bzw. START-CATALOG-MODIFICATION) soll geprüft werden.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Vollqualifizierter Name des statischen Subsystemkataloges, dessen Inhalt geprüft wird.

DEPENDENCE-RELATION = *YES / *NO
Vereinbart, ob in die Prüfung der Subsystemdefinitionen auch die verankerten Abhängigkeitsbeziehungen zu anderen Subsystemen einbezogen werden sollen (*YES) oder nicht (*NO). Der Katalog kann nicht gesichert werden, wenn in einer vorhergehenden CHECK-CATALOG-Anweisung *NO angegeben wurde.

LINK-RELATION = *YES / *NO
Vereinbart, ob in die Prüfung der Subsystemdefinitionen auch die verankerten Adressbeziehungen zu anderen Subsystemen einbezogen werden sollen (*YES, Voreinstellung) oder nicht (*NO). Der Katalog kann nicht gesichert werden, wenn in einer vorhergehenden CHECK-CATALOG-Anweisung *NO angegeben wurde.

RELATED-FILES = *NO / *YES
Bestimmt, ob die Existenz von Dateien, die in Abhängigkeitsbeziehungen zu diesen Subsystemen stehen, geprüft werden soll (*YES) oder nicht (*NO, Voreinstellung).
Wurden die Namen von abhängigen Dateien mit dem Wert *INSTALLED(...) definiert, wird auch der dort angegebene DEFAULT-NAME geprüft.

OUTPUT =
Vereinbart, wohin die von der Anweisung generierten Informationen - die Ergebnisse des Prüflaufs - ausgegeben werden sollen.

OUTPUT = *SYSOUT
Voreinstellung: die Meldungen werden auf der Datensichtstation ausgegeben.

OUTPUT = *SYSLST(...)
Die Meldungen sollen nach SYSLST ausgegeben werden.

SYSLST-NUMBER =
Bezeichnet die SYSLST-Datei, in die die Ausgabe erfolgen soll.

SYSLST-NUMBER = *STD
Voreinstellung: die Ausgabe soll in die Standard-Systemdatei SYSLST erfolgen.

SYSLST-NUMBER = <integer 1..99>
Die Ausgabe soll in eine Systemdatei aus der Menge SYSLST01 bis SYSLST99 erfolgen, deren Nummer hier anzugeben ist.

4.3.5 GENERATE-CATALOG-SOURCE

SSCM-Anweisungsliste für Generierung erstellen

Funktionsbeschreibung

Mit dieser Anweisung erstellt SSCM eine Datei mit einer Liste aller SSCM-Anweisungen, die für die (Wieder-) Generierung eines Subsystemkatalogs benötigt werden (für bestimmte oder für alle Subsysteme des Eingabe-Katalogs).

Format

GENERATE-CATALOG-SOURCE
CATALOG-NAME = ***CURR**ENT / <filename 1..54 without-gen-vers> ,**SUBSYSTEM-NAME** = ***ALL** / <structured-name 1..8>(...) <structured-name 1..8>(...) **VERSION** = ***ALL** / <c-string 3..8> / <text 3..8> ,**OUTPUT** = ***SYSLST** (...) ***SYSLST**(...) **SYSLST-NUM**BER = ***STD** / <integer 1..99>

Operandenbeschreibung

CATALOG-NAME =
Gibt den Subsystemkatalog an, in dem die Subsystemdefinitionen gespeichert sind.

CATALOG-NAME = *CURRENT
Voreinstellung. Es wird der aktuelle Subsystemkatalog verwendet.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Name des Subsystemkatalogs.

SUBSYSTEM-NAME =
Subsysteme, deren Definitionen ausgegeben werden sollen.

SUBSYSTEM-NAME = *ALL
Voreinstellung. Die Definitionen aller Subsysteme sollen ausgegeben werden.

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name des Subsystems, dessen Definition ausgegeben werden soll.

VERSION = *ALL / <c-string 3..8> / <text 3..8>
Version des Subsystems, dessen Definition ausgegeben werden soll.

OUTPUT = *SYSLST(...)
Systemdatei, an die die generierten Informationen geschickt werden sollen.

SYSLST-NUMBER = *STD
Voreinstellung. Die Informationen werden an die Systemdatei SYSLST geschickt.

SYSLST-NUMBER = <integer 1..99>

Die Informationen werden an die aus dem Bereich SYSLST01 bis SYSLST99 angegebene Systemdatei geschickt.

4.3.6 MODIFY-SUBSYSTEM-ATTRIBUTES

Eigenschaften von Subsystemen verändern

Funktionsbeschreibung

Mit dieser Anweisung können alle Eigenschaften und Einsprungstellen, die mit der Anweisung SET-SUBSYSTEM-ATTRIBUTES definiert wurden, verändert werden.

Bei der Veränderung einer Definition ist auf Folgendes zu achten:

- Das Subsystem - identifiziert durch Name und Version - muss im aktuell geöffneten Katalog gefunden werden
- Der Versuch, einen bereits bestehenden Auftragseingang oder bereits bestehende Beziehungen hinzuzufügen, wird abgewiesen
- Nicht zulässig ist, einen noch nicht bestehenden Auftragseingang oder noch nicht definierte Beziehungen zu verändern oder zu löschen
- Der Modus eines Auftragseingangs darf nur verändert werden, wenn alle Parameter angegeben werden; die Standardwerte ***UNCHANGED werden zurückgewiesen
- Die Speicherklasse eines Subsystems darf nur verändert werden, wenn alle Parameter angegeben werden; die Standardwerte ***UNCHANGED werden zurückgewiesen

MODIFY-SUBSYSTEM-ATTRIBUTES wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Hinweis zur Syntax

> Für die Namen der Einsprungstellen in den folgenden Operanden (im Format ist der Datentyp <name> angegeben) kann auch der spezielle Datentyp <symbol> verwendet werden, der im Handbuch „BLSSERV “ ausführlich beschrieben ist:
>
> - LINK-ENTRY
> - DYNAMIC-CHECK-ENTRY
> - INIT-ROUTINE
> - CLOSE-CTRL-ROUTINE
> - STOPCOM-ROUTINE
> - DEINIT-ROUTINE
> - INTERFACE-VERSION
> - ADD-SUBS-ENTRIES
> - MODIFY-SUBS-ENTRIES
> - REMOVE-SUBS-ENTRIES

Format

MODIFY-SUBSYSTEM-ATTRIBUTES
SUBSYSTEM-NAME = <structured-name 1..8>(...)

<structured-name 1..8>(...)

VERSION = <c-string 3..8> / <text 3..8>

,**INSTAL**LATION**-UNIT** = ***UNCHA** NGED / ***NONE** / ***STD** / <text 1..30>

,**INSTAL**LATION**-USERID** = ***UNCHA** NGED / ***NONE** / ***DEF**AULT**-USERID** / <name 1..8>

,**COPYRIGHT** = ***UNCHA** NGED / ***NONE** / <c-string 1..54>(...)

<c-string 1..54>(...)

YEAR = ***YEAR-1990** / <c-string 4..4>

,**LIB**RARY = ***UNCHA** NGED / ***STD** / ***CPLINK** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = ***** **UNCHA** NGED / <filename 1..30 without-catid-userid-gen-vers>

,**DEF**AULT**-NAME** = ***** **UNCHA** NGED / <filename 1..54 without-gen-vers>

,**SUBSYS**TEM**-LOAD-MODE** = ***UNCHA** NGED / ***STD** / ***ADVANCED**

,**REP-FILE** = ***UNCHA** NGED / ***STD** / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = ***** **UNCHA** NGED / <filename 1..30 without-catid-userid-gen-vers>

,**DEF**AULT**-NAME** = ***** **UNCHA** NGED / <filename 1..54 without-gen-vers> / ***NONE**

,**REP-FILE-MANDATORY** = ***UNCHA** NGED / ***NO** / ***Y**ES

,**MES**SAGE**-FILE** = ***UNCHA** NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = ***** **UNCHA** NGED / <filename 1..30 without-catid-userid-gen-vers>

,**DEF**AULT**-NAME** = ***** **UNCHA** NGED / <filename 1..54 without-gen-vers> / ***NONE**

,**SUBSYS**TEM**-INFO-FILE** = ***UNCHA** NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = ***** **UNCHA** NGED / <filename 1..30 without-catid-userid-gen-vers>

,**DEF**AULT**-NAME** = ***** **UNCHA** NGED / <filename 1..54 without-gen-vers> / ***NONE**

,**SYNTAX-F**ILE = ***UNCHA** NGED / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = ***** **UNCHA** NGED / <filename 1..30 without-catid-userid-gen-vers>

,**DEF**AULT**-NAME** = ***** **UNCHA** NGED / <filename 1..54 without-gen-vers> / ***NONE**

,**DYNAMIC-CHECK-ENTRY** = ***UNCHA** NGED / ***STD** / ***NO** / <text 1..8 without-sep>

,**CRE**ATION**-TIME** = ***UNCHA** NGED / ***AT-CRE**ATION**-REQ**UEST / ***AT-SUBSYS**TEM**-CALL**(...) /

***AT-DSSM-LOAD** / ***BEFORE-DSSM-LOAD** / ***MANDATORY-AT-STARTUP** /

***BEFORE-SYSTEM-READY** / ***AFTER-SYSTEM-READY**

***AT-SUBSYS**TEM**-CALL**(...)

ON-ACTION = ***STD** / ***ISL-CALL** / *****ANY

,**INIT-ROUTINE** = ***UNCHA** NGED / ***NO** / <text 1..8 without-sep>

,**CLOSE-CTRL-ROUTINE** = ***UNCHA** NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOPCOM-ROUTINE** = ***UNCHA** NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**DEINIT-ROUTINE** = ***UNCHA** NGED / ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOP-AT-SHUTDOWN** = ***UNCHA** NGED / ***NO** / ***Y**ES

,**INTERFACE-VERSION** = ***UNCHA** NGED / ***NO** / <text 1..8 without-sep>

,**SUBSYS**TEM**-HOLD** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBID**DEN

,**STATE-CHANGE-CMDS** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBID**DEN / ***BY-ADMINISTRATOR-ONLY**

,**FORCED-STATE-CHANGE** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBID**DEN

,**RESET** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBID**DEN

,**RESTART-REQUIRED** = ***UNCHA** NGED / ***NO** / ***Y**ES

,**VERSION-COEXISTENCE** = ***UNCHA** NGED / ***FORBID**DEN / ***ALLOW**ED

,**VERSION-EXCHANGE** = ***UNCHA** NGED / ***FORBID**DEN / ***ALLOW**ED

,**ADD-SUBS-ENTRIES** = ***NONE** / list-poss(100): <text 1..8>(...)

<text 1..8>(...)

MODE = ***LINK** / ***ISL**(...) / ***SVC**(...) / ***SYSTEM-EXIT**(...)

***ISL**(...)

FUNCTION-NUMBER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SVC**(...)

NUMBER = <integer 0..255>

,**CALL-BY-SYSTEM-EXIT** = ***ALLOW** ED / ***FORBIDDEN**

,**FUNCTION-NUM**BER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SYSTEM-EXIT**(...)

NUMBER = <integer 0..127>

,**CONNECTION-ACCESS** = ***ALL** / ***SYS**TEM / ***SIH**

,**CONNECTION-SCOPE** = ***TASK** / ***PROG**RAM / ***FREE** / ***CALL** / ***OPTIMAL**

,**FIRST-CONNECTION** = ***ALLOW** ED / ***FORBIDDEN**

,**MODIFY-SUBS-ENTRIES** = ***NONE** / list-poss(100): <text 1..8>(...) **/ *BY-PROGRAM(...)**

<text 1..8>(...)

MODE = ***UNCHA** NGED / ***LINK** / ***ISL**(...) / ***SVC**(...) / ***SYSTEM-EXIT**(...)

***ISL**(...)

FUNCTION-NUMBER = ***UNCHA** NGED (...) / ***NONE** / <integer 0..255>(...)

***UNCHA**NGED(...)

FUNCTION-VERSION = ***UNCHA** NGED / <integer 1..255>

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SVC**(...)

NUMBER = ***UNCHA** NGED / <integer 0..255>

,**CALL-BY-SYSTEM-EXIT** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBIDDEN**

,**FUNCTION-NUMBER** = ***UNCHA** NGED (...) / ***NONE** / <integer 0..255>(...)

***UNCHA**NGED(...)

FUNCTION-VERSION = ***UNCHA** NGED / <integer 1..255>

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SYSTEM-EXIT**(...)

NUMBER = <integer 0..127>

,**CONNECTION-ACCESS** = ***UNCHA** NGED / ***ALL** / ***SYS**TEM / ***SIH**

,**CONNECTION-SCOPE** = ***UNCHA** NGED / ***TASK** / ***PROG**RAM / ***FREE** / ***CALL** / ***OPTIMAL**

,**FIRST-CONNECTION** = ***UNCHA** NGED / ***ALLOWED** / ***FORBIDDEN**

***BY-PROGRAM(...)**

CONNECTION-SCOPE = ***UNCHA** NGED / ***TASK** / ***PROGRAM**

,**REMOVE-SUBS-ENTRIES** = ***NONE** / list-poss(100): <text 1..8>

,**MEMORY-CLASS** = ***UNCHA** NGED / ***SYSTEM-GLOBAL**(...) / ***LOCAL-PRIVILEGED**(...) /

***LOCAL-UNPRIVILEGED**(...) **/ *BY- SLICE**(...)

***SYSTEM-GLOBAL**(...)

SUBSYSTEM-**ACCESS** = ***LOW** / ***SYS**TEM / ***HIGH**

***LOCAL-PRIVILEGED**(...)

SIZE = <integer 1..32767>

***LOCAL-UNPRIVILEGED**(...)

SIZE = ***UNCHA** NGED / <integer 1..32767>

,**SUBSYS**TEM**-ACCESS** = ***UNCHA** NGED / ***LOW** / ***HIGH**

,**START-ADDRESS** = ***UNCHA** NGED / ***ANY** / <x-string 7..8>

***BY- SLICE**(...)

SIZE = **<integer** 1..32767**>**

,**LINK-ENTRY** = ***UNCHA** NGED / <text 1..8 without-sep>(...)

<text 1..8 without-sep>(...)

AUTOLINK = ***ALLOW** ED / ***FORBIDDEN**

,**ADD-REFER-SUBS** = ***NONE** / list-poss(15): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***LOW** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***HIGH** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**MODIFY-REFER-SUBS** = ***NONE** / list-poss(15): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***UNCHA** NGED / ***LOW**EST**-EXIST**ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***UNCHA** NGED / ***HIGH**EST**-EXIST**ING / <c-string 3..8> / <text 3..8>

,**REMOVE-REFER-SUBS** = ***NONE** / list-poss(15): <structured-name 1..8>

,**UNRESOLVED-EXTERNALS** = ***UNCHA** NGED / ***ALLOW**ED / ***FORBIDDEN**

,**CHECK-REFERENCE** = ***UNCHA** NGED / ***Y**ES / ***NO**

,**ADD-RELATED-SUBS** = ***NONE** / list-poss(100): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***LOW** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***HIGH** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**MODIFY-RELATED-SUBS** = ***NONE** / list-poss(100): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***UNCHA** NGED / ***LOW**EST**-EXIST**ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***UNCHA** NGED / ***HIGH**EST**-EXIST**ING / <c-string 3..8> / <text 3..8>

,**REMOVE-RELATED-SUBS** = ***NONE** / list-poss(100): <structured-name 1..8>

Operandenbeschreibung

Der Standardwert *UNCHANGED bedeutet jeweils, dass der in der Anweisung SET-SUBSYSTEM-ATTRIBUTES eingestellte Wert gültig bleiben soll.
Wird der Typ des vereinbarten Auftragseingangs (Operand MODE) oder der subsystemspezifische Adressraum (Operand MEMORY) verändert, müssen alle Unteroperanden von MODE bzw. MEMORY explizit mit einem Wert versehen werden (der Operandenwert *UNCHANGED (Voreinstellung) wird abgewiesen).

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Vereinbart Name und Version des Subsystems, dessen Definition verändert werden soll.

VERSION = <c-string 3..8> / <text 3..8>
Die Version des Subsystems ist im Format „[V][m]m.n[aso]“ zu vereinbaren, wobei die Textteile folgende Bedeutung besitzen:

mm: Hauptversion (numerisch)

n: Nachtragsversion (numerisch)

aso: Änderungsstand (a: Buchstabe, Freigabestand; so: numerisch, Korrekturstand)

INSTALLATION-UNIT =
Legt den Namen der installierten Liefergruppe fest. Für alle mit IMON zu installierenden Subsysteme muss ein Wert ungleich *NONE angegeben werden, ebenso, wenn bei den Operanden SUBSYSTEM-LIBRARY, REP-FILE, SUBSYSTEM-INFO-FILE, MESSAGE-FILE und SYNTAX-FILE der Wert *INSTALLED(LOGICAL-ID=...) definiert wurde.

Die im Handbuch „IMON“ dargestellten Syntaxregeln sind bei der Festlegung des Namens zu beachten.

INSTALLATION-UNIT = *NONE
Es wird kein Name vergeben. Für alle mit IMON installierten Subsysteme ist diese Angabe nicht erlaubt.

INSTALLATION-UNIT = *STD
Der beim Operanden SUBSYSTEM-NAME angegebene Name wird als neuer Name der installierten Liefergruppe genutzt.

INSTALLATION-UNIT = <text 1..30>
Neuer Name der installierten Liefergruppe.

INSTALLATION-USERID =
Vereinbart eine Benutzerkennung, unter der die zuständige DSSM-Task die Nebenkomponenten des Subsystems (Rep-Datei, Objektmodulbibliothek, Meldungsdatei, Syntaxdatei und Subsystem-Informationsdatei) erwartet, falls diese Dateien noch keiner Benutzerkennung zugeordnet sind, d.h. der Dateiname ohne Benutzerkennung angegeben wurde.

INSTALLATION-USERID = *NONE
Die Dateien werden nicht unter einer bestimmten Benutzerkennung erwartet.

INSTALLATION-USERID = *DEFAULT-USERID
Die Dateien werden unter der System-Standardkennung erwartet (Prefix „$.“) bzw. unter der Benutzerkennung der aufrufenden Task, wenn es ein lokales Subsystem ist.

INSTALLATION-USERID = <name 1..8>
Benutzerkennung, unter der die Nebenkomponenten erwartet werden.
Gilt die Anweisung für ein SSD-Object, werden die Dateien nur dann unter der hier angegebenen Benutzerkennung erwartet, wenn in der Anweisung ADD-CATALOG-ENTRY (Übernahme von Subsystemdefinitionen aus dem SSD-Object in den Katalog) keine Benutzerkennung angegeben wurde. Die Angabe bei ADD-CATALOG-ENTRY hat Vorrang.

COPYRIGHT =
Vereinbart, ob und welche Copyright-Meldung beim Starten des Subsystems ausgegeben werden soll.

COPYRIGHT = *NONE
Es soll keine Copyright-Meldung ausgegeben werden.

COPYRIGHT = <c-string 1..54>(...)
Text der Copyright-Meldung, die zusammen mit dem Datum der Erstellung beim Starten ausgegeben wird.

> **YEAR = *YEAR-1990 / <c-string 4..4>**
> Jahreszahl, die in der Copyright-Meldung als Datum der Erstellung erscheinen soll. Eine semantische Prüfung findet nicht statt.

LIBRARY =
Vereinbart einen neuen Namen für die Programm- oder Bindemodulbibliothek (OML), aus der der Objectcode des Subsystems bei dessen Aktivierung geladen werden soll.

LIBRARY = *STD
Der Objectcode wird beim Starten automatisch aus der Bibliothek SYSLNK.<subsysname>.<subsysvers#> geladen. Sie ist auf der Benutzerkennung abgelegt, unter der die Holdertask läuft; also auf der Benutzerkennung des Aufrufers bei lokalen Subsystemen und auf TSOS bei globalen Subsystemen.

Der Wert von „subsysvers#“ ist dreistellig und setzt sich aus den beim Operanden SUBSYSTEM-NAME=... (VERSION=...) angegebenen Teilen „mmn“ zusammen.

LIBRARY = *CPLINK
Das zu definierende Subsysteme ist mit dem Basissystem des BS2000 verknüpft (CP=Control Program) und muss bereits vor der Aktivierung von DSSM geladen sein.
Der Operand darf nur in Verbindung mit dem Operanden CREATION-TIME=*BEFORE-DSSM-LOAD verwendet werden.

LIBRARY = *INSTALLED(...)
Der Bibliotheksname muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.

Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

> **LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>**
> Logischer Name der Programm- oder Bindemodulbibliothek.
>
> **DEFAULT-NAME = <filename 1..54 without-gen-vers>**
> Bibliotheksname bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

LIBRARY = <filename 1..54 without-gen-vers>
Vollqualifizierter Dateiname der Bindemodulbibliothek, aus der der Objectcode für das Subsystem geladen werden soll.

SUBSYSTEM-LOAD-MODE =
Bestimmt den Lademodus des Subsystems (über die BLS-DSSM-Schnittstelle $PBBND1).

SUBSYSTEM-LOAD-MODE = *STD
Das BLS wird im STD-Run-Mode aufgerufen und lädt das Subsystem als Objektmodul.

SUBSYSTEM-LOAD-MODE = *ADVANCED
Das BLS wird im ADVANCED-Run-Mode aufgerufen und lädt das Subsystem als Bindelademodul (LLM).

REP-FILE =
Legt fest, ob System-Reps für das zu definierende Subsystem benötigt werden und in welcher Datei diese hinterlegt sind. Diese Korrekturanweisungen werden während der Aktivierung des Subsystems ausschließlich auf die in der Bindemodulbibliothek hinterlegten und geladenen Module angewandt, nicht auf andere Subsysteme oder BS2000-CP. Eine Rep-Datei kann auch für Module eines nicht-privilegierten Subsystems vereinbart werden.

REP-FILE darf nicht zusammen mit LIBRARY=*CPLINK angegeben werden.

REP-FILE = *STD
Standardmäßig werden die System-Reps aus der Rep-Datei mit dem Namen

SYSREP.<subsysname>.<subsysvers#> geladen. Diese Datei ist auf der Benutzerkennung abgelegt, unter der die Holdertask läuft; also auf der Benutzerkennung des Aufrufers bei lokalen Subsystemen und auf TSOS bei globalen Subsystemen.
Der Wert von „subsysvers#“ ist dreistellig und setzt sich aus den beim Operanden SUBSYSTEM-NAME=... (VERSION=...) angegebenen Teilen „mmn“ zusammen.

REP-FILE = *NO
Für das Subsystem soll keine Rep-Datei verarbeitet werden.

REP-FILE = *INSTALLED(...)
Der Name der Rep-Datei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.

Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

> **LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>**
> Logischer Name der Rep-Datei.
>
> **DEFAULT-NAME =**
> Name der Rep-Datei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.
>
> **DEFAULT-NAME = <filename 1..54 without-gen-vers>**
> Es wird ein neuer Name vergeben.
>
> **DEFAULT-NAME = *NONE**
> Es wird kein neuer Name vergeben.

REP-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Rep-Datei, aus der die Korrekturanweisungen gelesen werden.

REP-FILE-MANDATORY =
Legt fest, ob eine mit dem Operanden REP-FILE deklarierte Rep-Datei beim Laden des Subsystems abgearbeitet werden muss oder nicht.

REP-FILE-MANDATORY = *NO
Der Einsatz einer Rep-Datei ist nicht Pflicht, d.h. weder die Rep-Datei noch deren Einträge sollen beim Aktivieren des Subsystems geprüft werden. Sollte die Rep-Datei nicht zugreifbar oder einzelne Korrekturanweisungen fehlerhaft sein, wird das Subsystem auch in diesem Fall gestartet.

REP-FILE-MANDATORY = *YES
Sollte es bei der Bearbeitung der Rep-Datei zu folgenden Fehlern kommen, wird der Versuch, das Subsystem zu laden, abgebrochen:

- Die Rep-Datei ist nicht katalogisiert oder kann nicht gelesen werden
- Die Prüfung der Korrekturanweisungen zeigt einen Fehler an
- Der Name der Korrekturanweisungen ist fehlerhaft
- Das DMS meldet einer Fehler beim Zugriff auf die NOREF-Datei (diese Datei wird während des Ladens eines Subsystems benutzt, um zu verhindern, dass ungültige System-Reps an der Bedienstation protokolliert werden)

MESSAGE-FILE =
Bestimmt, ob es eine subsystemspezifische Meldungsdatei gibt, die beim Laden des Subsystems automatisch aktiviert wird.

Für Subsysteme, die mit dem Startzeitpunkt AT-DSSM-LOAD definiert werden, ist im Operanden RELATED-SUBSYSTEM eine Abhängigkeitsbeziehung zum Subsystem MIP zu vereinbaren.

MESSAGE-FILE = *NO
Es soll keine Meldungsdatei aktiviert werden. Dieser Wert ist verpflichtend für alle Subsysteme, die mit dem Startzeitpunkt BEFORE-DSSM-LOAD definiert werden (siehe auch Operand CREATION-TIME), da zu diesem Zeitpunkt noch keine Meldungsdatei aktiviert werden kann.

MESSAGE-FILE = *INSTALLED(...)
Der Name der Meldungsdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

> **LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>**
> Logischer Name der Meldungsdatei.
>
> **DEFAULT-NAME =**
> Name der Meldungsdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.
>
> **DEFAULT-NAME = <filename 1..54 without-gen-vers>**
> Es wird ein neuer Name vergeben.
>
> **DEFAULT-NAME = *NONE**
> Es wird kein neuer Name vergeben.

MESSAGE-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Meldungsdatei. Diese wird beim Laden des Subsystems (Kommando START-SUBSYSTEM) automatisch aktiviert, beim Entladen des Subsystems (Kommando STOP-SUBSYSTEM) automatisch deaktiviert.

SUBSYSTEM-INFO-FILE =
Bestimmt, ob eine Subsysteminformationsdatei (SSINFO) vorhanden ist. In dieser Datei sind subsystemspezifische Daten (Nebenkomponenten und Konfigurationsdaten) enthalten, die nicht von DSSM zentral bearbeitet werden können.

SUBSYSTEM-INFO-FILE = *NO
Eine Informationsdatei für das Subsystem ist nicht verfügbar.

SUBSYSTEM-INFO-FILE = *INSTALLED(...)
Der Name der Informationsdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.

Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logischer Name der Informationsdatei.

DEFAULT-NAME =
Name der Informationsdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Es wird ein neuer Name vergeben.

DEFAULT-NAME = *NONE
Es wird ein neuer Name vergeben.

SUBSYSTEM-INFO-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Informationsdatei. Der Name wird automatisch an die Aktivierungs- und Deaktivierungsroutinen (Operanden INIT-/DEINIT-/CLOSE-CTRL-ROUTINE) übergeben, wenn diese aufgerufen werden.

SYNTAX-FILE =
Bestimmt, ob eine Syntaxdatei mit dem Subsystem verknüpft ist, die beim Laden des Subsystems automatisch aktiviert wird. Für Subsysteme, die mit dem Startattribut MANDATORY-AT-STARTUP definiert werden, ist im Operanden RELATED-SUBSYSTEM eine Abhängigkeitsbeziehung zum Subsystem SDF zu deklarieren.

SYNTAX-FILE = *NO
Es soll keine Syntaxdatei aktiviert werden. Dieser Wert ist verpflichtend für alle Subsysteme, die mit dem Startzeitpunkt BEFORE-DSSM-LOAD oder AT-DSSM-LOAD definiert werden (siehe auch Operand CREATION-TIME), da zu diesem Zeitpunkt noch keine Syntaxdatei aktiviert werden kann.

SYNTAX-FILE = *INSTALLED(...)
Der Name der Syntaxdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logischer Name der Syntaxdatei.

DEFAULT-NAME =
Name der Syntaxdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Es wird ein neuer Name vergeben.

DEFAULT-NAME = *NONE
Es wird kein neuer Name vergeben.

SYNTAX-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Syntaxdatei, die beim Laden des Subsystems automatisch aktiviert werden soll.

DYNAMIC-CHECK-ENTRY =
Vereinbart, ob eine dynamische Identitätsprüfung des Subsystems vorgenommen werden soll. Zu diesem Zweck muss eine Einsprungstelle angegeben werden, an der sowohl der Subsystemname (acht Zeichen) als auch die Versionsnummer (vier bzw. sieben Zeichen) stehen muss. DSSM prüft, ob die bei der Definition vergebene Identifikation mit dem geladenen Subsystem übereinstimmt.

DYNAMIC-CHECK-ENTRY = *STD
Es soll gelten, dass die bei dem Operanden LINK-ENTRY spezifizierte Einsprungstelle für die Identitätsprüfung herangezogen werden soll.

DYNAMIC-CHECK-ENTRY = *NO
Eine Überprüfung soll nicht stattfinden. Dieser Operandenwert darf allerdings für solche Subsysteme, die vor der Aktivierung von DSSM geladen sein sollen (CREATION-TIME=*BEFORE-DSSM-LOAD), nicht verwendet werden.

DYNAMIC-CHECK-ENTRY = <text 1..8 without-sep>
Name der Einsprungstelle, die für die Identitätsprüfung herangezogen werden soll.

CREATION-TIME =
Legt den Zeitpunkt fest, an dem die Aktivierung des Subsystems (CREATE-Routine) angestoßen wird.
Während der Systemeinleitung sind zwei Phasen zu unterscheiden, in denen DSSM nach Aufruf durch die Startup-Routine die Steuerung der Systemeinleitung übernimmt:

Phase 1: Der DSSM-Code wird geladen, die DSSM-Task generiert und gestartet.

Die Task reserviert Klasse-5-Speicher, liest den Subsystemkatalog ein und startet die Subsysteme, die mit den Startattributen BEFORE-DSSM-LOAD und AT-DSSM-LOAD definiert wurden.

Nach dem Laden dieser Subsysteme geht die Steuerung der Systemeinleitung an die Startup-Routine zurück.

Phase 2: Nach erneutem Aufruf werden alle Subsysteme geladen, die mit den Startattributen MANDATORY-AT-STARTUP, BEFORE-SYSTEM-READY und AFTER-SYSTEM-READY definiert wurden.

Bei den beiden erstgenannten wird das Laden der Subsysteme mit der Startup-Routine synchronisiert (d.h. das Laden muss abgeschlossen sein), beim letztgenannten wird das Laden asynchron angestoßen.

Die Steuerung der Systemeinleitung geht an die Startup-Routine zurück.

Sollen verschiedene Versionen eines Subsystems definiert werden, können die Startattribute, die für die Phasen 1 und 2 der Systemeinleitung vorgesehen sind, nur für eine dieser Versionen vergeben werden.

CREATION-TIME = *AT-CREATION-REQUEST
Das Subsystem muss explizit mit dem Kommando START-SUBSYSTEM geladen werden.

CREATION-TIME = *AT-SUBSYSTEM-CALL(...)
Das Subsystem soll automatisch beim ersten SVC- oder ISL-Aufruf geladen werden. Dieser Operandenwert ist reserviert für Subsysteme, die über SVC oder ISL aufgerufen werden.
Sind mehrere Versionen eines Subsystems mit diesem Operandenwert definiert, muss für alle diese Versionen VERSION-COEXISTENCE=*ALLOWED angegeben werden sowie FUNCTION-NUMBER und FUNCTION-VERSION für ihre SVC- bzw. ISL-Einsprungstellen, die mit CONNECTION-ACCESS ungleich *SIH deklariert wurden.

Mindestens eines der angegebenen Subsysteme muss mit SUBSYSTEM-ENTRIES ...,MODE=*SVC oder *ISL deklariert worden sein (übereinstimmend mit dem Wert des Operanden ON-ACTION).

ON-ACTION =
Bestimmt, wodurch das automatische Laden des Subsystems veranlasst wird.

ON-ACTION = *STD
Voreinstellung: das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden SVC-Einsprungstelle.

ON-ACTION = *ISL-CALL
Das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden ISL-Einsprungstelle.

ON-ACTION = *ANY
Das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden SVC- oder ISL-Einsprungstelle.

CREATION-TIME = *AT-DSSM-LOAD
Das Subsystem soll während der Systemeinleitung (Phase 1) unter der Kontrolle der DSSM-Task geladen werden. Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die ebenfalls mit diesem Startattribut definiert sind oder das Startattribut BEFORE-DSSM-LOAD besitzen.

Die Dateien für dieses Subsystem müssen unter der Benutzerkennung TSOS auf dem Home-Pubset angelegt sein, da zum Startzeitpunkt weder der Benutzerkatalog zugreifbar, noch die IMPORT-PUBSET-Verarbeitung abgeschlossen ist.
Für diese Subsysteme ist das Einbinden einer Syntaxdatei nicht zulässig.

CREATION-TIME = *BEFORE-DSSM-LOAD
Das Subsystem soll während der Systemeinleitung (Phase 1), aber nicht unter der Kontrolle der DSSM-Task geladen werden.
Solche Subsysteme sind mit dem Organisationsprogramm verknüpft und brauchen - bei der Aktivierung - nicht mit der DSSM-Task synchronisiert werden. Nach dem Laden des Subsystems läuft dieses allerdings wieder unter der Kontrolle von DSSM und kann aus Sicht des Anwenders wie andere Subsysteme gesteuert werden.
Adress- oder Abhängigkeitsbeziehungen zu Subsystemen, die mit einem anderen Startattribut definiert wurden, sind nicht möglich. Auch die Einbindung einer Meldungs- oder Syntaxdatei ist nicht zulässig. Alle Auftragseingänge (Operand SUBSYSTEM-ENTRIES) müssen deklariert sein, da DSSM die (privilegierte) Verbindung zu diesen Auftragseingängen herstellt. Es liegt in der Verantwortung des Subsystem-Entwicklers, sicherzustellen, dass zu jedem Zeitpunkt mindestens eine Version dieses Subsystems verfügbar ist.
Der Name des Link-Kontextes für diese Subsysteme muss eindeutig sein, da DSSM eine Entlade-Anforderung erfüllen muss, selbst wenn DSSM den Subsystem-Code nicht geladen hat. Eine Einsprungstelle (Operand DYNAMIC-CHECK-ENTRY) muss angegeben sein.

CREATION-TIME = *BEFORE-SYSTEM-READY
Das Subsystem soll während der Systemeinleitung (Phase 2) geladen werden. Die Aktivierung wird synchron angestoßen; die Steuerung geht erst nach dem vollständigen Laden (oder nach Lade-Fehler) an die Startup-Routine zurück, die dann „SYSTEM READY“ melden kann.
Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD, AT-DSSM-LOAD oder MANDATORY-AT-STARTUP definiert wurden.
Die Dateien für dieses Subsystem müssen auf dem Home-Pubset katalogisiert sein.
Wird ein nicht-privilegiertes Subsystem mit diesem Operandenwert deklariert, bekommt es implizit den Wert *AFTER-SYSTEM-READY zugewiesen. SSCM gibt eine Meldung aus.

CREATION-TIME = *MANDATORY-AT-STARTUP
Das Subsystem muss während der Systemeinleitung (Phase 2) geladen werden. Die Aktivierung wird - wie bei BEFORE-SYSTEM-READY - synchron angestoßen. Im Unterschied zum oben genannten muss das Laden des Subsystems allerdings **erfolgreich** abgeschlossen werden. Andernfalls geht eine Meldung an die Startup-Routine, dass ein verpflichtendes Subsystem nicht geladen werden konnte. Die Startup-Routine entscheidet in diesem Fall, ob die Verarbeitung fortgesetzt oder abgebrochen wird.

Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD oder AT-DSSM-LOAD definiert wurden. Die Dateien für dieses Subsystem müssen auf dem Home-Pubset liegen.

CREATION-TIME = *AFTER-SYSTEM-READY
Das Laden des Subsystems wird während der Systemeinleitung (Phase 2) angestoßen. Die Durchführung dieses Auftrags wird nicht mit der Startup-Routine synchronisiert, die vor dem Abschluss des Ladens „SYSTEM READY“ melden kann.
Das Subsystem darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD, AT-DSSM-LOAD, MANDATORY-AT-STARTUP oder BEFORE-SYSTEM-READY definiert wurden. Die Dateien für dieses Subsystem müssen auf dem Home-Pubset liegen.

INIT-ROUTINE =
Legt fest, ob eine Initialisierungsroutine für das Subsystem durchlaufen werden soll, wenn es gestartet oder fortgesetzt wird. In diesem Fall muss der Name einer Einsprungstelle bekannt sein und DSSM delegiert die Initialisierung an die Holdertask des betreffenden Subsystems.
Für alle Subsysteme mit dem Startattribut BEFORE-DSSM-LOAD wird die Angabe einer Einsprungstelle unbedingt empfohlen. Beim Laden des Subsystems (d.h. dem Durchlaufen der Initialisierungsroutine) erhält dieses dann Kenntnis davon, dass DSSM die Kontrolle über Anschluss und Abbau von Beziehungen übernehmen kann.

INIT-ROUTINE = *NO
Es soll keine Initialisierungsroutine durchlaufen werden.

INIT-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der Initialisierungsroutine.
Der Initialisierungsroutine wird in der Holdertask die Steuerung übergeben, damit sich das Subsystem initialisieren kann. Dazu werden ihr übergeben:

- der Name und die Version des Subsystems, wie im SSMCAT definiert
- der Name der SSINFO-Datei, falls dieser im Operanden SUBSYSTEM-INFO-FILE spezifiziert wurde
- die Adresse der beim Laden und Binden angegebenen Einsprungstelle (LINK-ENTRY)
- der vom Dynamischen Bindelader verwendete Binder-Kontext-Name

- der Name des Memory-Pools (für Subsysteme im Klasse-5- oder Klasse-6-Speicher), damit sich das Subsystem beim Nachladen eigener Selectable Units/Load Units darauf beziehen kann
- der Name der Meldungsdatei
- die Adresse des Operanden SUBSYSTEM-PARAMETER, falls im Kommando START-SUBSYSTEM ein String angegeben wird

Am Ende der Initialisierung wird eine Rückmeldung des Subsystems erwartet, ob die Initialisierung erfolgreich durchgeführt wurde und ob die Holdertask als Arbeitstask genutzt werden soll (wird in der Anweisung ASSIGN-HOLDER-TASK vereinbart). Abhängig davon steht die Task dann weiter unter DSSM-Kontrolle oder unter Kontrolle des Subsystems.

CLOSE-CTRL-ROUTINE =
Legt fest, ob eine Routine in das Subsystem eingebunden ist, die das Anhalten/Deaktivieren des Subsystems steuert.
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).
Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE. Beim Ansprung dieser Routine wird sichergestellt, dass noch Verbindung zum Subsystem besteht.

Existiert eine CLOSE-CTRL-Routine, tritt bei einem Versionswechsel während der BS2000-Session keine Unterbrechung auf. Es existiert zu jedem Zeitpunkt genau eine gültige Version (entweder die alte Version ist noch verfügbar oder die neue Version ist bereits verfügbar). Ohne eine solche Routine beinhaltet ein Versionswechsel immer eine Anschlussunterbrechung, während die STOPCOM-Routine der alten Version und die INIT-Routine der neuen Version ablaufen (siehe dazu auch "Austausch von Subsystemversionen").

CLOSE-CTRL-ROUTINE = *NO
Im betreffenden Subsystem ist keine Routine verankert, die das Deaktivieren oder Anhalten des Subsystems steuert.

CLOSE-CTRL-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse oder FITC-Port. Die notwendigen Parameter werden vom Subsystem am Ende der INIT-Routine dynamisch an die CLOSE-CTRL-Routine übergeben und DSSM über die Identifikation der Börse bzw. FITC-Ports in Kenntnis gesetzt.
Voraussetzung für die Nutzung der CLOSE-CTRL-Routine sind die Angabe einer INIT-Routine (Operand INIT-ROUTINE) und einer STOPCOM-Routine (Operand STOPCOM-ROUTINE= *NO/*DYNAMIC).

Beim Aufruf der CLOSE-CTRL-Routine muss die Holdertask des Subsystems als Arbeitstask genutzt werden (Anweisung ASSIGN-HOLDER-TASK).

CLOSE-CTRL-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

STOPCOM-ROUTINE =
Legt fest, ob eine Routine in das Subsystem eingebunden ist, die das aktive Beenden der Aufträge durchführen kann.

STOPCOM-ROUTINE = *NO
Im betreffenden Subsystem ist keine Routine verankert.

STOPCOM-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse oder FITC. Die notwendigen Parameter werden vom Subsystem am Ende der CLOSE-CTRL-Routine oder (wenn eine solche nicht vorhanden ist) am Ende der INIT-Routine dynamisch an die STOPCOM-Routine übergeben. DSSM wird über die Identifikation der Börse bzw. FITC-Ports in Kenntnis gesetzt.Voraussetzung für die Nutzung der STOPCOM-Routine ist die Angabe einer INIT-Routine (Operand INIT-ROUTINE). Beim Aufruf der STOPCOM-Routine muss die Holdertask des Subsystems als Arbeitstask genutzt werden (siehe ASSIGN-HOLDER-TASK).

STOPCOM-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

DEINIT-ROUTINE =
Legt fest, ob eine Routine in das Subsystem eingebunden ist, die die Deinitialisierung des Subsystems durchführen kann. Diese Deinitialisierungsroutine realisiert die Rückgabe der vom Subsystem angeforderten Betriebsmittel (Speicher, Dateien, Geräte).
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).
Wird ein Subsystem mit einer INIT- und einer CLOSE-CTRL-Routine definiert, muss eine DEINIT-Routine - mit dem gleichen Operandenwert wie die CLOSE-CTRL-Routine - angegeben werden. Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE. Beim Ansprung dieser Routine ist sichergestellt, dass aufrufende Tasks nicht mehr an das Subsystem angeschlossen werden und alle vorhandenen Aufruf-Beziehungen gelöst werden.

DEINIT-ROUTINE = *NO
Im betreffenden Subsystem ist keine Deinitialisierungsroutine verankert, die die Rückgabe der Betriebsmittel veranlasst; diesen Part übernimmt DSSM selbst.

DEINIT-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse bzw. FITC.
Die notwendigen Parameter werden vom Subsystem am Ende der STOPCOM-Routine oder (wenn eine solche nicht vorhanden ist) am Ende der CLOSE-CTRL-Routine oder am Ende der INIT-Routine, wenn weder eine STOPCOM- noch eine CLOSE-CTRL-Routine eingebunden ist, dynamisch an die DEINIT-Routine übergeben.
DSSM wird über die Identifikation der Börse bzw. des FITC-Ports in Kenntnis gesetzt.
Voraussetzung für die Nutzung der DEINIT-Routine ist die Angabe einer INIT-Routine (Operand INIT-ROUTINE).
Beim Aufruf der DEINIT-Routine muss die Holdertask des Subsystems als Arbeitstask genutzt werden (Anweisung ASSIGN-HOLDER-TASK).

DEINIT-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

STOP-AT-SHUTDOWN =
Legt fest, ob das Subsystem nach Beendigung der Benutzertasks bei Shutdown automatisch entladen werden soll.

STOP-AT-SHUTDOWN = *NO
Das Subsystem wird nicht automatisch entladen.
Die Angabe sollte nicht für Subsysteme verwendet werden, die Adressbeziehungen zu anderen Subsystemen besitzen, die mit STOP-AT-SHUTDOWN=*YES definiert sind.

STOP-AT-SHUTDOWN = *YES
Das Subsystem wird bei Shutdown automatisch entladen.
Diese Angabe wird ignoriert, wenn keine STOPCOM-, DEINIT- oder CLOSE-CTRL-Routine angegeben wird.
SSCM gibt eine Meldung aus.

INTERFACE-VERSION =
Bezeichnet die Einsprungstelle, über die DSSM auf diejenige Schnittstellenversion zugreifen kann, die für den Aufruf der INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routinen benutzt werden soll.

INTERFACE-VERSION = *NO
Das Subsystem ruft keine INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routine auf.

INTERFACE-VERSION = <text 1..8 without-sep>
Name der Einsprungstelle. Die Einsprungstelle verweist auf den Standardheader, in dem u.a. auch die Schnittstellenversion hinterlegt ist. Der Standardheader wird durch den Aufruf des Makros $ESMINT(I) mit MF=I/L generiert.
Dieser Operand ist Pflicht für Subsysteme, für die eine INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routine definiert wurde.

SUBSYSTEM-HOLD =
Legt fest, ob das geladene Subsystem angehalten oder entladen werden kann.

SUBSYSTEM-HOLD = *ALLOWED
Das geladene Subsystem kann angehalten und entladen werden. Die Kommandos HOLD-

SUBSYSTEM und STOP-SUBSYSTEM sind für dieses Subsystem zulässig.

SUBSYSTEM-HOLD = *FORBIDDEN
Die Kommandos HOLD-SUBSYSTEM und STOP-SUBSYSTEM sind für dieses Subsystem nicht zulässig; das Subsystem wird - entsprechend den Angaben im Operanden STOP-AT-

SHUTDOWN - bei Shutdown entladen. Wird das Subsystem durch Austausch mit einem anderen Subsystem entladen, so erfolgt der Austausch unterbrechungsfrei.

STATE-CHANGE-CMDS =
Legt fest, ob die DSSM-Kommandos zur Steuerung des Subsystems im laufenden Betrieb (START-SUBSYSTEM, STOP-SUBSYSTEM, HOLD-SUBSYSTEM, RESUME-SUBSYSTEM) verwendet werden dürfen.

Wird von einer Version eines Subsystems in eine andere gewechselt, wird der bei STATE-CHANGE-CMDS angegebene Wert für die auszuwechselnde Version nicht berücksichtigt.

STATE-CHANGE-CMDS = *ALLOWED
Die Kommandos dürfen an der Bedienstation und unter der privilegierten Benutzerkennung (Benutzerkennung, die mit dem Systemprivileg SUBSYSTEM-MANAGEMENT ausgestattet ist) verwendet werden.

STATE-CHANGE-CMDS = *FORBIDDEN
Die Kommandos dürfen generell nicht - weder an der Bedienstation noch unter der privilegierten Benutzerkennung - verwendet werden.

STATE-CHANGE-CMDS = *BY-ADMINISTRATOR-ONLY
Die Kommandos dürfen nur unter der privilegierten Benutzerkennung verwendet werden; für den Operator an der Bedienstation sind die Kommandos gesperrt.
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).
Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE. Beim Ansprung dieser Routine ist sichergestellt, dass aufrufende Tasks nicht mehr an das Subsystem angeschlossen werden. Tasks, die noch in Aufruf-Beziehung zum Subsystem stehen, bleiben davon unberührt.

FORCED-STATE-CHANGE =
Legt fest, ob innerhalb der Kommandos STOP-SUBSYSTEM und HOLD-SUBSYSTEM die Verwendung des Operanden FORCED=*YES zulässig ist. Mit dieser Funktion kann das unbedingte Deaktivieren des Subsystems erzwungen werden.

FORCED-STATE-CHANGE = *FORBIDDEN
Das Deaktivieren des Subsystems kann nicht erzwungen werden. DSSM weist die Verwendung des Operanden FORCED in den entsprechenden Kommandos mit einer Fehlermeldung zurück.

FORCED-STATE-CHANGE = * ALLOWED
Die Verwendung des Operanden FORCED=*YES für dieses Subsystem ist zulässig. Dieser Operandenwert darf nicht zusammen mit SUBSYSTEM-HOLD=*FORBIDDEN angegeben werden.

RESET =
Legt fest, ob innerhalb der Kommandos START-SUBSYSTEM und RESUME-SUBSYSTEM die Verwendung des Operanden RESET=*YES zulässig ist. Mit dieser Funktion kann das unbedingte Laden bzw. Fortsetzen des Subsystems erzwungen werden, auch wenn sich das Subsystem im Zustand IN-DELETE bzw. IN-HOLD befindet.

RESET = *FORBIDDEN
Das Aktivieren des Subsystems kann nicht erzwungen werden. DSSM weist die Verwendung des Operanden RESET in den entsprechenden Kommandos mit einer Fehlermeldung zurück.

RESET = *ALLOWED
Die Verwendung des Operanden RESET=*YES für dieses Subsystem ist zulässig.

Dieser Operandenwert darf nicht mit SUBSYSTEM-HOLD=*FORBIDDEN angegeben werden.

RESTART-REQUIRED =
Legt fest, ob bei abnormaler Beendigung der Holdertask die Initialisierungsroutine für das Subsystem durchlaufen werden soll.

RESTART-REQUIRED = *NO
Die Initialisierungsroutine wird für einen Wiederanlauf des Subsystems nicht benutzt.

RESTART-REQUIRED = *YES
Die Initialisierungsroutine soll bei abnormaler Beendigung der Holdertask benutzt werden. Voraussetzung ist, dass die Durchführung dieser Routine im Operanden INIT-ROUTINE vorgesehen ist.

VERSION-COEXISTENCE =
Vereinbart, ob mehr als eine Version des gleichen Subsystems gleichzeitig aktiv sein darf.

VERSION-COEXISTENCE = *FORBIDDEN
Die aktuelle Version des Subsystems kann nicht gleichzeitig mit einer anderen Version des gleichen Subsystems koexistieren.

VERSION-COEXISTENCE = *ALLOWED
Die aktuelle Version des Subsystems kann gleichzeitig mit einer anderen Version des gleichen Subsystems koexistieren (Coexistence-Modus).
Bei der Definition des Auftragseingangs (Operand SUBSYSTEM-ENTRIES) darf keine indirekte Verbindung über System-Exit-Routinen gewählt werden. Sind verschiedene Versionen des gleichen Subsystems geladen, die mit dem gleichen Auftragseingang definiert wurden, wird der Anschluss immer an die höchste geladene Version des Subsystems realisiert.
Greifen koexistente Subsysteme auf koexistente Syntaxdateien zu, müssen diese im SSD-Object deklariert sein und können nicht von SDF verwaltet werden.
Bei Anschlüssen über SVC und ISL ist jedoch eine Versionsauswahl über die Operanden FUNCTION-NUMBER und FUNCTION-VERSION möglich.

VERSION-EXCHANGE =
Vereinbart, ob das Laden der aktuellen Subsystemversion im Exchange-Modus erlaubt ist. Der Exchange-Modus erlaubt die temporäre Koexistenz zweier Versionen des gleichen Subsystems. Wird die Version B eines Subsystems geladen, wenn bereits Version A des Subsystems aktiv ist, werden alle neuen Aufrufer an die Version B angeschlossen. Die Aufträge, die an die Version A angeschlossen sind, werden noch bearbeitet. Nach Bearbeitung aller Aufträge an Version A wird diese automatisch beendet.
Bei der Definition ist darauf zu achten, dass die zu ersetzende, „alte“ Version nicht abhängig von der ersetzenden, „neuen“ Version sein darf.

VERSION-EXCHANGE = *FORBIDDEN
Die aktuelle Version des Subsystems darf nicht ausgetauscht werden.

VERSION-EXCHANGE = *ALLOWED
Der Exchange-Modus, der die temporäre Koexistenz zweier Subsysteme erlaubt, ist für die aktuelle Subsystemversion zulässig.

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES =
Bezeichnet, ob neue Auftragseingänge definiert (ADD) oder die Charakteristiken vorhandener Auftragseingänge verändert (MODIFY) werden sollen.

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES = *NONE
Voreinstellung: es sollen weder neue Auftragseingänge hinzukommen, noch die Eigenschaften vorhandener Auftragseingänge modifiziert werden.

ADD-SUBS-ENTRIES / MODIFY-SUBS-ENTRIES = list-poss(100): <text 1..8>
Vereinbart durch Angabe des Namens der Einsprungstelle entweder maximal 100 neue Auftragseingänge für das Subsystem, deren Typ jeweils in den Unterstrukturen definiert werden muss (ADD), oder modifiziert bereits definierte Auftragseingänge (MODIFY).

MODE =
Legt den Typ eines vereinbarten Auftragseingangs für das Subsystem fest.
Wird der Typ des vereinbarten Auftragseingangs verändert, müssen alle Unteroperanden von MODE explizit mit einem Wert versehen werden; der Operandenwert *UNCHANGED (Voreinstellung bei MODIFY-SUBS-ENTRIES) wird abgewiesen.

MODE = *LINK
Der Auftragseingang kann nicht über indirektes Linkage erreicht werden, sondern nur über eine CONNECT-Beziehung mittels eines externen Binder-Symbols.
Bei verschiedenen Versionen des gleichen Subsystems, die das gleiche externe Binder-Symbol nutzen, stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.

MODE = *ISL(...)
Der Auftragseingang wird durch indirekte Verbindung über System Procedure Linkage (nur für privilegierte Subsysteme) erreicht. Wird zusätzlich noch eine Funktions- und Versionsnummer der ISL-Einsprungstelle spezifiziert, darf sich die Kombination aus Name der Einsprungstelle, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED) decken.

Bei ungleichen Subsystemen, deren Auftragseingang über die selbe ISL-Einsprungstelle erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die die selbe ISL-Einsprungstelle nutzen (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die die selbe ISL-Einsprungstelle nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist, ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standard Header der Parameterliste des Aufrufers hinterlegt ist. Die Angabe CONNECTION-ACCESS=*ALL ist für ISL-Einsprungstellen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der ISL-Einsprungstelle angesprochen werden soll, da die gleiche ISL-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der ISL-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten ISL-Funktionsnummer.

MODE = *SVC(...)
Der Auftragseingang wird durch indirekte Verbindung über Supervisor Call (SVC) erreicht.

Wird zusätzlich noch eine Funktions- und Versionsnummer der SVC-Einsprungstelle spezifiziert, darf sich die Kombination aus SVC-Nummer, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED) decken.
Bei ungleichen Subsystemen, deren Auftragseingang über den selben SVC erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist, ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standard Header der Parameterliste des Aufrufers hinterlegt ist.

NUMBER = <integer 0..255>
Nummer des SVCs, über den der Auftragseingang erreicht wird. In Verbindung mit CONNECTION-ACCESS=*ALL ist die Verwendung einer SVC-Nummer größer 191 nicht zulässig.

CALL-BY-SYSTEM-EXIT =
Legt fest, ob die angegebene SVC-Nummer von System-Exit-Routinen aus aufgerufen werden darf.

CALL-BY-SYSTEM-EXIT = *ALLOWED
Der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen zulässig.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
Der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der SVC-Einsprungstelle angesprochen werden soll, da die gleiche SVC-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der SVC-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten SVC-Funktionsnummer.

MODE = SYSTEM-EXIT(...)
Der Auftragseingang wird durch indirekte Verbindung über System-Exit-Routinen erreicht.
In Verbindung mit CONNECTION-ACCESS=*ALL ist die Verwendung dieses Operanden nicht zulässig.

NUMBER = <integer 0..127>
Nummer der System-Exit-Routine.

CONNECTION-ACCESS =
Vereinbart die Zugriffsberechtigung (Privilegierung) für das Subsystem.

CONNECTION-ACCESS = *ALL
Privilegierte und nicht-privilegierte Programmläufe sind zugriffsberechtigt.
In Verbindung mit MODE=*SYSTEM-EXIT/*ISL/*SVC (mit einer SVC-Nummer größer 191) ist die Verwendung dieses Operandenwertes nicht zulässig.

CONNECTION-ACCESS = *SYSTEM
Nur privilegierte Programmläufe sind zugriffsberechtigt.

CONNECTION-ACCESS = *SIH
Nur Tasks, die im Funktionszustand SIH laufen, sind zugriffsberechtigt.
Das aufgerufene Subsystem läuft ebenfalls im Funktionszustand SIH, d.h. es ist nicht unterbrechbar.
Die Angabe dieses Operandenwertes ist nur für Subsysteme zulässig, deren Auftragseingang definiert wird über

- System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
- CONNECTION-SCOPE=*OPTIMAL
- MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Bezeichnet das Ereignis, das die automatische Auflösung des Anschlusses an den angegebenen Subsystem-Auftragseingang hervorruft.

CONNECTION-SCOPE = *TASK
Der Anschluss wird bei Taskbeendigung aufgehoben.

CONNECTION-SCOPE = *PROGRAM
Der Anschluss wird spätestens bei Programmbeendigung aufgehoben. Zusammen mit MEMORY-CLASS=*LOCAL-UNPRIVILEGED darf nur CONNECTION-SCOPE=*PROGRAM angegeben werden.

Die Angabe dieses Operandenwertes wird für Subsysteme empfohlen, die mit SUBSYSTEM-ACCESS=*LOW/*HIGH oder MEMORY-CLASS=*BY-SLICE deklariert wurden.

CONNECTION-SCOPE = *FREE
DSSM soll keine Kontrolle von Anschlüssen zu diesen Auftragseingängen durchführen. Der Anschluss wird - außer bei einer expliziten Anforderung - nicht automatisch aufgelöst. Um Probleme oder mögliche Fehler beim Entladen des Subsystems zu vermeiden, müssen die Anschlüsse vom Subsystem selbst verwaltet werden.

CONNECTION-SCOPE = *CALL
Nach Rückkehr aus diesem Auftragseingang führt DSSM automatisch die Auflösung der Anschlüsse durch. Dieser Operandenwert steht nur für Subsysteme zur Verfügung, deren Auftragseingang über System Procedure Linkage (ISL) und Supervisor Call (SVC) definiert wird.

CONNECTION-SCOPE = *OPTIMAL
Das Subsystem wird deaktiviert bzw. angehalten, wenn keine Task mehr Anschluss zu diesem Auftragseingang hat.
Eine Routine, deren Einsprungstelle mit *OPTIMAL definiert wird, muss mit RETURN beendet werden.
Wird ein Auftragseingang eines Subsystems mit CONNECTION-SCOPE=*OPTIMAL definiert, müssen alle seine Auftragseingänge im Subsystemkatalog mit MODE !=*LINK definiert werden.
Während ein Subsystem deaktiviert oder angehalten wird, wird kein Aufruf des Subsystems mit CONNECTION-SCOPE=*OPTIMAL akzeptiert.

FIRST-CONNECTION =
Bestimmt, ob der Erst-Anschluss der Task an den angegebenen Auftragseingang des Subsystems erlaubt ist. Mindestens ein Auftragseingang eines Subsystems muss mit FIRST-CONNECTION=*ALLOWED definiert werden.

FIRST-CONNECTION = *ALLOWED
Der Erst-Anschluss an den angegebenen Auftragseingang ist erlaubt.
Für die Anweisung ADD-SUBS-ENTRIES ist dieser Wert die Voreinstellung.

FIRST-CONNECTION = *FORBIDDEN
Der Anschluss an den angegebenen Auftragseingang über SVC oder ISL ist nicht erlaubt, wenn die Task nicht bereits an einen anderen Auftragseingang des Subsystems angeschlossen ist.
Die Angabe dieses Operandenwertes ist für Auftragseingänge, die mit MODE=*LINK/*SYSTEM-EXIT oder CONNECTION-ACCESS=*SIH definiert wurden, nicht erlaubt.

MODIFY-SUBS-ENTRIES = *NONE / list-poss(100): <text 1..8> / *BY-PROGRAM(...)
Die Werte *NONE und list-poss(100): <text 1..8> sind beim Operanden ADD-SUBS-ENTRIES beschrieben.

MODIFY-SUBS-ENTRIES = *BY-PROGRAM(...)
Die Einsprungstellen des angegebenen Subsystems werden nicht statisch aus dem Katalog versorgt, sondern dynamisch zum Ladezeitpunkt aus dem BLS-Namensverzeichnis. Voraussetzung für diese Funktionalität ist der Einsatz von BLSSERV ab Version 2.1, das den Einsatz von EEN-Namen als Einsprungstellen für DSSM-Subsysteme unterstützt.

Die Anweisung wird zurückgewiesen, wenn das Subsystem nicht zuvor bei der Anweisung SET-SUBSYSTEM-ATTRIBUTES auch mit *BY-PROGRAM definiert wurde. MODIFY-SUBS-ENTRIES dient nur zum Ändern der Anschluss-Einstellungen.

Wird *BY-PROGRAM verwendet, müssen die Operanden ADD-SUBS-ENTRIES und REMOVE-SUBS-ENTRIES mit *NONE angegeben werden.

CONNECTION-SCOPE = *TASK / *PROGRAM
Der Anschluss wird bei Task- bzw. Programmbeendigung aufgehoben.

REMOVE-SUBS-ENTRIES =
Bestimmt, ob vorhandene Auftragseingänge, die für das Subsystem vereinbart sind, gelöscht werden sollen.

REMOVE-SUBS-ENTRIES = *NONE
Voreinstellung: Es sollen keine Auftragseingänge gelöscht werden.

REMOVE-SUBS-ENTRIES = list-poss(100): <text 1..8>
Gibt die Namen von maximal 100 Auftragseingängen an, die für das Subsystem nicht mehr gelten sollen.

MEMORY-CLASS =
Vereinbart den subsystemspezifischen Adressraum, in den das Subsystem geladen werden soll. Mit diesem Operanden kann die Systembetreuung die Adressraumvorgaben, die für das jeweilige Subsystem gelten, den speziellen Anforderungen der Installation anpassen.
Wird der subsystemspezifische Adressraum verändert, müssen alle Unteroperanden von

MEMORY explizit mit einem Wert versehen werden (der Operandenwert *UNCHANGED (Voreinstellung) wird abgewiesen).

MEMORY-CLASS = *SYSTEM-GLOBAL(...)
Das Subsystem wird in den Klasse-3- oder Klasse-4-Speicher geladen. Residente CSECTs erhalten Klasse-3-Speicher, alle anderen erhalten seitenwechselbaren Klasse-4-Speicher.

SUBSYSTEM-ACCESS =
Bezeichnet die Zugriffsrechte (Privilegierung) und die Lage des angeforderten Speichers.

SUBSYSTEM-ACCESS = *LOW
Es wird nicht-privilegierter Adressraum unterhalb der 16-MByte-Grenze zugewiesen.

SUBSYSTEM-ACCESS = *SYSTEM
Subsysteme, die mit diesem Operandenwert deklariert werden, sind privilegierte Subsysteme, denen privilegierter Adressraum oberhalb der 16-MByte-Grenze zugewiesen wird.

Die Angabe dieses Operandenwertes ist für die Subsysteme verpflichtend, deren Auftragseingang über SVC (MODE=*SVC), oder für die eine INIT-, STOPCOM-, DEINIT- oder CLOSE-CTRL-Routine vereinbart wird. Der Operandenwert ist unzulässig in Kombination mit CONNECTION-ACCESS=*ALL und MODE=*LINK.

SUBSYSTEM-ACCESS = *HIGH
Es wird nicht-privilegierter Adressraum bis zu 2 GByte zugewiesen.

MEMORY-CLASS = *LOCAL-PRIVILEGED(...)
Das Subsystem erhält einen Memory Pool im nicht-privilegierten Klasse-5-Speicher, der unterhalb der 16-MByte-Grenze angelegt wird.
Die Angabe ist auf nicht-privilegierte Subsysteme zugeschnitten, die relativ viel Adressraum (+/- 1 MByte) beanspruchen und unterhalb der 16-MByte-Grenze anzulegen sind. Die Subsysteme werden in Memory Pools an der gleichen Adresse geladen, um mit dem knapp bemessenen Adressraum unterhalb 16-MByte zu haushalten. Obwohl solche Subsysteme parallel im gleichen Adressraum geladen werden, können sie nicht simultan von einer Task genutzt werden (siehe auch Anweisung SEPARATE-ADDRESS-SPACE).
Das Subsystem darf keine residenten CSECTs enthalten, da ansonsten ein späteres Aktivieren abgebrochen wird.

SIZE = <integer 1..32767>
Größe des benötigten Adressraums (in 4KByte-Seiten) für den Memory Pool im Klasse-5-Speicher. Der Wert ist mindestens so groß zu wählen, dass das Subsystem und evtl. von diesem nachgeladene Selectable Units /Load Units in vollem Umfang geladen werden können. Die obere Grenze ist generierungsabhängig.

MEMORY-CLASS = *LOCAL-UNPRIVILEGED(...)
Das Subsystem erhält einen Memory Pool im nicht-privilegierten Klasse-6-Speicher. Die Angabe ist für Subsysteme reserviert, die wie ein Programm ausführbar sind. Analog dazu muss deren Zugriffsberechtigung (Privilegierung) im Operanden CONNECTION-ACCESS mit dem Wert *ALL definiert werden.
Dieser Operandenwert darf nicht zusammen mit einer Einsprungstelle angegeben werden, die mit CONNECTION-ACCESS=*SYSTEM definiert wurde.
Das Subsystem darf keine residenten CSECTs enthalten, da ansonsten ein späteres Aktivieren abgebrochen wird.
Wird dieser Operandenwert angegeben, ist zur Auflösung des Anschlusses an den angegebenen Subsystem-Auftragseingang nur CONNECTION-SCOPE=*PROGRAM erlaubt.

SIZE = <integer 1..32767>
Größe des benötigten Adressraums (in 4KByte-Seiten) für den Memory Pool im Klasse-6-Speicher. Der Wert ist mindestens so groß zu wählen, dass das Subsystem und evtl. von diesem nachgeladene Selectable Units /Load Units in vollem Umfang geladen werden können. Die obere Grenze ist generierungsabhängig.

SUBSYSTEM-ACCESS =
Bezeichnet die Lage des angeforderten Speichers.

SUBSYSTEM-ACCESS = *LOW
Es wird nicht-privilegierter Adressraum unterhalb der 16-MByte-Grenze zugewiesen. Da diese Angabe auf Subsysteme zugeschnitten ist, die wie Programme ausführbar sind, ist die zusätzliche Angabe CONNECTION-SCOPE=*PROGRAM anzuraten.

SUBSYSTEM-ACCESS = *HIGH
Es wird nicht-privilegierter Adressraum bis zu 2 GByte zugewiesen.
Da diese Angabe auf Subsysteme zugeschnitten ist, die wie Programme ausführbar sind, ist die zusätzliche Angabe CONNECTION-SCOPE=*PROGRAM anzuraten.

START-ADDRESS =
Legt die Anfangsadresse im Klasse-6-Speicher fest.

START-ADDRESS = *ANY
Die Lage des Subsystems im Klasse-6-Speicher wird von DSSM festgelegt.

START-ADDRESS = <x-string 7..8>
Start-Adresse im Segment-Raster, an der die Anfangsadresse des Subsystems liegen soll. Als Wert ist eine 8-stellige Hexadezimalkonstante anzugeben, die ein Vielfaches von X'100000' sein muss.

MEMORY-CLASS = *BY-SLICE(...)
Das angegebene Subsystem ist ein nicht-privilegiertes Subsystem und besteht aus einem LLM, das aus einem mehrbenutzbaren Code (Programmbereich) und einem nichtmehrbenutzbaren Code (Datenbereich) besteht.
Der Programmbereich wird in den gemeinsam benutzbaren Adressraum geladen (das entspricht MEMORY-CLASS=*SYSTEM-GLOBAL). Der Datenbereich wird in den Benutzeradressraum der Holdertask geladen und in die privaten Benutzeradressräume der angeschlossenen Tasks an die selbe Adresse kopiert.
Gemeinsam mit MEMORY-CLASS=*BY-SLICE müssen folgende Werte angegeben werden:

SUBSYSTEM-LOAD-MODE=*ADVANCED und CONNECTION-ACCESS=*ALL.

SIZE = <integer 1..32767>
Größe des angeforderten Speicherplatzes für den Datenbereich in 4K-Seiten.
Der Wert ist mindestens so groß zu wählen, dass der Datenbereich und evtl. vom Subsystem in den Benutzeradressraum nachzuladende Selectable Units/Load Units in vollem Umfang geladen werden können.
Die obere Grenze ist generierungsabhängig.

LINK-ENTRY = <text 1..8 without-sep>(...)
Definiert den Namen des zum Laden benötigten Bindemoduls/ENTRY/CSECT (als Operand im Makroaufruf $PBBND1 an den dynamischen Bindelader DBL). Das Subsystem muss von diesem ENTRY vollständig (ggf. per Autolink) geladen werden.

AUTOLINK =
Steuert den Aufruf der Autolink-Funktion beim Binden und Laden.
Die Autolink-Funktion des Binders ermöglicht das automatische Einfügen von Modulen, die nicht mit entsprechenden Anweisungen explizit eingefügt werden. Die Funktion erspart vor allem den Benutzern der höheren Programmiersprachen, die zahlreich benötigten Module des Laufzeitsystems (= Run Time System) mit expliziten Anweisungen einzeln einzufügen. Ein nähere Beschreibung der Autolink-Funktion ist im Handbuch „BLSSERV“ zu finden.
Die Autolink-Funktion kann auch implizit umgangen werden, wenn während des Bindens des zu ladenden Objektmoduls der erste Externverweis auf ein vorgebundenes Großmodul zielt. Der Vorteil dieser Vorgehensweise ergibt sich daraus, dass das Seitenwechselverhalten bei der späteren Ausführung bereits im Vorfeld (während des Bindens) optimiert werden kann. Zudem können Fehler während des Bindens auf diese Weise vermieden werden.

AUTOLINK = *ALLOWED
Die Autolink-Funktion wird zugelassen.

AUTOLINK = *FORBIDDEN
Die Autolink-Funktion wird unterdrückt.

ADD-REFER-SUBS / MODIFY-REFER-SUBS =
Legt fest, ob entweder eine Liste von Subsystemen anzulegen ist, zu denen Adressbeziehungen bestehen und die zur Auflösung von Externverweisen benutzt werden sollen (ADD), oder ob eine bereits bestehende Liste existiert, die zu verändern ist (MODIFY).

ADD-REFER-SUBS / MODIFY-REFER-SUBS = *NONE
Voreinstellung: es soll weder eine Liste angelegt, noch eine bereits bestehende Liste verändert werden.

ADD-REFER-SUBS / MODIFY-REFER-SUBS = list-poss(15):<structured-name 1..8>
Es sollen Externverweise angegeben (ADD) bzw. modifiziert (MODIFY) werden.
Es können Externverweise auf maximal 15 andere Subsysteme benannt werden, die zur Auflösung dieser Externverweise benutzt werden müssen. Fehlt eines der hier genannten Subsysteme beim Aktivieren oder Deaktivieren (und ist gleichzeitig eine Überprüfung der Externverweise mit dem Operanden CHECK-REFERENCE=*YES angefordert), wird die Aktion abgebrochen.
Auch das „Basis-Subsystem“ des BS2000, das Control Programm, kann über diese Externverweise - mit dem Namen CP - angesprochen werden. Es kann in der folgenden Unterstruktur sowohl genau eine Version spezifiziert werden, als auch ein Bereich von Versionen angegeben werden, innerhalb dessen alle Versionen herangezogen werden sollen.

Wird die Version des Subsystems CP über eine Versionsbereichsliste eingegrenzt, prüft DSSM die Kompatibilität zwischen der aktuellen CP-Version und Versionen in der Bereichsliste. Nur wenn die Versionen kompatibel sind, wird das Subsystem geladen.

Auf folgende Einschränkungen bei der Angabe von Subsystemen, zu denen Adressbeziehungen bestehen, ist zu achten:

- Es dürfen keine Adressbeziehungen zu Subsystemen vereinbart werden, die das Attribut MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED/*BY-SLICE besitzen.
- Ist für das zu definierende Subsystem das Attribut SUBSYSTEM-ACCESS=*SYSTEM vergeben, dürfen keine Subsysteme angesprochen werden, die mit SUBSYSTEM-ACCESS=*LOW bzw. *HIGH definiert sind.
- Subsysteme, die das Attribut STOP-AT-SHUTDOWN=*YES besitzen, dürfen Adressbeziehungen nur zu solchen Subsystemen aufweisen, die ebenfalls dieses Attribut besitzen.
- Ein nicht-privilegiertes Subsystem darf generell keine Adressbeziehung zum Control Programm CP aufweisen.
- Wird auf ein Subsystem verwiesen, das zumindest eine Version besitzt, die im Coexistence- oder Exchange-Modus betrieben werden darf, ist die eindeutige Version anzugeben.
- Die Adressbeziehungen müssen in Abhängigkeit der Startattribute (Operand CREATION-TIME) definiert werden, d.h. Subsysteme dürfen Beziehungen nur zu solchen Subsystemen aufnehmen, deren Start zeitgleich oder früher erfolgt.

LOWEST-VERSION =
Spezifiziert den unteren Wert (niedrigste Version) innerhalb der Versionsbereichsliste der Subsysteme.

LOWEST-VERSION = *LOWEST-EXISTING
Die niedrigste Version im Katalog soll angesprochen werden.

LOWEST-VERSION = <c-string 3..8> / <text 3..8>
Version eines Subsystems, die als Untergrenze des Versionsbereiches fungieren soll.

HIGHEST-VERSION =
Spezifiziert den oberen Wert (höchste Version) innerhalb der Versionsbereichsliste der Subsysteme.

HIGHEST-VERSION = *HIGHEST-EXISTING
Die höchste Version im Katalog soll angesprochen werden.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>
Version eines Subsystems, die als Obergrenze des Versionsbereiches fungieren soll.

MODIFY-REFER-SUBS =
Siehe Beschreibung bei ADD-REFER-SUBS.

REMOVE-REFER-SUBS =
Bestimmt, ob bestehende Externverweise auf andere Subsysteme gelöscht werden sollen.

REMOVE-REFER-SUBS = *NONE
Voreinstellung: Es sollen keine bestehenden Externverweise auf andere Subsysteme gelöscht werden.

REMOVE-REFER-SUBS = list-poss(15): <structured-name 1..8>
Gibt die Namen von maximal 15 Externverweisen an, die nicht mehr gültig sein sollen.

UNRESOLVED-EXTERNALS =
Vereinbart das Verhalten des Ladevorgangs, wenn Externverweise nicht aufgelöst werden können.

UNRESOLVED-EXTERNALS = *FORBIDDEN
Der Ladevorgang wird bei nicht auflösbaren Externverweisen abgebrochen.

UNRESOLVED-EXTERNALS = *ALLOWED
Der Ladevorgang wird fortgesetzt, nicht auflösbare Externverweise werden mit dem Wert X'FFFFFFFF' besetzt.

CHECK-REFERENCE =
Vereinbart, ob die im Operanden REFERENCED-SUBSYSTEM angegebenen Subsysteme im Hinblick auf deren Zustand und Verfügbarkeit hin überprüft werden sollen.

CHECK-REFERENCE = *YES
Die Referenz-Subsysteme werden überprüft. Fehlt eines der Subsysteme, bricht DSSM das Aktivieren oder Entladen des Subsystems ab.

CHECK-REFERENCE = *NO
DSSM soll keine Prüfung vornehmen.
Generiert der Anwender allerdings mit dieser Anweisung auch komplexe Subsysteme, führt DSSM die geforderten Funktionen **trotz** möglicher Konflikte durch:

- Das Kommando START-SUBSYSTEM lädt das angegebene Subsystem, auch wenn ein Subsystem, zu dem definierte Beziehungen bestehen, noch nicht vollständig geladen ist.
- Die Kommandos RESUME-SUBSYSTEM, STOP-SUBSYSTEM und HOLD-SUBSYSTEM werden ohne Prüfung von Beziehungen und Abhängigkeiten von DSSM ausgeführt.

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS =
Legt fest, ob entweder eine Liste von Subsystemen aufgebaut werden soll, zu denen Abhängigkeitsbeziehungen bestehen (ADD), oder eine bereits bestehende Liste abhängiger Subsysteme modifiziert werden soll (MODIFY).

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS = *NONE
Voreinstellung: es sollen für das Subsystem weder Abhängigkeitsbeziehungen definiert noch verändert werden.

ADD-RELATED-SUBS / MODIFY-RELATED-SUBS = list-poss(100):<structured-name 1..8>
Es sollen Abhängigkeitsbeziehungen zu maximal 100 anderen Subsystemen, ohne die das zu definierende Subsystem nicht funktionsfähig ist, definiert (ADD) oder verändert (MODIFY) werden.

Abhängigkeitsbeziehungen dürfen auch zum BS2000 Control Program CP definiert werden. Die Regeln und Einschränkungen, die hierbei zu beachten sind, gelten analog für Adressbeziehungen und sind dort näher erläutert (siehe Operand ADD-REFER-SUBS).
Die Abhängigkeitsbeziehungen zielen jeweils auf eine Version eines Subsystems. In der folgenden Unterstruktur kann sowohl genau eine Version spezifiziert werden, als auch ein Bereich von Versionen angegeben werden, innerhalb dessen alle Versionen herangezogen werden sollen.

Auf folgende Einschränkungen ist allgemein bei der Angabe von abhängigen Subsystemen ist zu achten:

- Die definierte Beziehung muss zyklenfrei sein. Ein Zyklus entsteht, wenn Subsystem A abhängig von B, B abhängig von C und dieses wieder abhängig von A ist.
- Ist für das zu definierende Subsystem das Attribut MEMORY-CLASS=*SYSTEM-GLOBAL vergeben, dürfen keine Subsysteme angesprochen werden, die mit MEMORY-CLASS=*LOCAL-PRIVILEGED oder *LOCAL-UNPRIVILEGED definiert sind.
- Für Subsysteme, die das Attribut SUBSYSTEM-ACCESS=*SYSTEM besitzen, darf keine Abhängigkeitsbeziehung zu Subsystemen definiert werden, für die SUBSYSTEM-ACCESS=*LOW bzw. SUBSYSTEM-ACCESS=*HIGH oder MEMORY-CLASS=*BY-SLICE gilt.
- Die Abhängigkeitsbeziehungen müssen entsprechend den Startattributen (Operand CREATION-TIME) definiert werden, d.h. ein Subsystem darf nur von solchen Subsystemen abhängen, deren Start zeitgleich oder früher erfolgt.

LOWEST-VERSION =
Spezifiziert den unteren Wert (niedrigste Version) innerhalb der Versionsbereichsliste der Subsysteme.

LOWEST-VERSION = *LOWEST-EXISTING
Die niedrigste Version im Katalog soll angesprochen werden.

LOWEST-VERSION = <c-string 3..8> / <text 3..8>
Version eines Subsystems, die als Untergrenze des Versionsbereiches fungieren soll.

HIGHEST-VERSION =
Spezifiziert den oberen Wert (höchste Version) innerhalb der Versionsbereichsliste der Subsysteme.

HIGHEST-VERSION = *HIGHEST-EXISTING
Die höchste Version im Katalog soll angesprochen werden.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>
Version eines Subsystems, die als Obergrenze des Versionsbereiches fungieren soll.

MODIFY-RELATED-SUBS =
Siehe Beschreibung bei ADD-RELATED-SUBS.

REMOVE-RELATED-SUBS =
Bestimmt, ob bestehende Abhängigkeitsbeziehungen zu anderen Subsystemen gelöscht werden sollen.

REMOVE-RELATED-SUBS = *NONE
Voreinstellung: Es sollen keine bestehenden Abhängigkeitsbeziehungen zu anderen Subsystemen gelöscht werden.

REMOVE-RELATED-SUBS = list-poss(100): <structured-name 1..8>
Gibt die Namen von maximal 100 Subsystemen an, zu denen keine Abhängigkeitsbeziehungen mehr bestehen sollen.

4.3.7 MODIFY-WORK-TASK-ATTRIBUTE

Arbeitstask-Parameter verändern

Funktionsbeschreibung

Mit dieser Anweisung können Subsysteme angegeben werden, die ihre Holdertask nicht mehr als Arbeitstask nutzen sollen. Zudem kann eine neue TSN angegeben werden, die die Holdertask erhalten soll. Die Anweisung wird zurückgewiesen, wenn das angegebene Subsystem nicht existiert oder die Holdertask nicht als Arbeitstask nutzt.

MODIFY-WORK-TASK-ATTRIBUTE wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Format

MODIFY-WORK-TASK-ATTRIBUTE
SUBSYSTEM-NAME = <structured-name 1..8>
,**SUBSYS**TEM**-VERS**ION = <c-string 3..8> / <text 3..8>
,**WORK-TASK** = ***UNCHA**NGED (...) / ***NO**
***UNCHA**NGED(...)
TSN = ***BY-DSSM** / <alphanum-name 1..4>

Operandenbeschreibung

SUBSYSTEM-NAME = <structured-name 1..8>
Name des Subsystems, das die Holdertask als Arbeitstask nutzt.

SUBSYSTEM-VERSION = <c-string 3..8> / <text 3..8>
Version des Subsystems, das die Holdertask als Arbeitstask nutzt.
Die Version muss bereits deklariert sein.

WORK-TASK =
Legt fest, ob die Holdertask nicht mehr als Arbeitstask genutzt werden soll oder ob lediglich die TSN für die Holdertask neu vergeben wird.

WORK-TASK = *UNCHANGED
Das Subsystem soll die Holdertask nach wie vor als Arbeitstask benutzen; in der folgenden Unterstruktur kann jedoch eine neue TSN für die Holdertask vergeben werden.

TSN =
Bestimmt, welche Task Sequence Number (TSN) die Holdertask des Subsystems erhalten soll.

TSN = *BY-DSSM
Voreinstellung: die TSN wird beim Laden der Arbeitstask von DSSM vergeben.

TSN = <alphanum-name 1..4>

TSN, die die Holdertask beim Starten erhalten soll.

Die angegebene TSN muss eindeutig festgelegt und verwendbar sein, wenn das Subsystem geladen wird.

WORK-TASK = *NO

Die Holdertask des angegebenen Subsystems soll nicht mehr als Arbeitstask verwendet werden.

4.3.8 REMOVE-ADDR-SPACE-SEPARATION

Disjunkte Verteilung der Subsysteme im Klasse-5-Speicher aufheben

Funktionsbeschreibung

Mit dieser Anweisung können die Vereinbarungen, die mit der Anweisung SEPARATE-ADDRESS-SPACE bezüglich der disjunkten Verteilung der Subsysteme im Klasse-5-Speicher getroffen wurden, aufgehoben werden.

Die Angabe ist für Subsysteme im Klasse-3- oder Klasse-4-Speicher (Operand MEMORY-CLASS=*SYSTEM-GLOBAL in der Anweisung SET-SUBSYSTEM-ATTRIBUTES) sowie für Subsysteme im Klasse-6-Speicher (MEMORY-CLASS=*LOCAL-UNPRIVILEGED) nicht relevant.

Subsysteme im Klasse-6-Speicher werden nie parallel gebraucht, dürfen sich also damit im Adressraum überschneiden, und Subsysteme im Klasse-3- oder Klasse-4-Speicher überschneiden sich grundsätzlich nicht.

REMOVE-ADDR-SPACE-SEPARATION wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Der Name des Subsystems muss im Katalog bekannt sein. Besteht eine in der Liste der Subsysteme angegebene Adressbeziehung nicht, wird die Angabe ignoriert; die Verarbeitung der Anweisung wird fortgesetzt.

Format

REMOVE-ADDR-SPACE-SEPARATION
SUBSYSTEM-NAME = <structured-name 1..8> ,**FROM-SUBSYSTEMS** = list-poss(15): <structured-name 1..8>

Operandenbeschreibung

SUBSYSTEM-NAME = <structured-name 1..8>
Name des Subsystems, das sich vormals nicht mit anderen Subsystemen überschneiden durfte.

Das Subsystem muss SSCM bekannt und bereits definiert sein.

FROM-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
Liste von maximal 15 Subsystemen, für die die Vereinbarung besteht, dass sie sich nicht mit dem im Operanden SUBSYSTEM-NAME angeführten Subsystem überschneiden dürfen. Die Vereinbarung wird für diese Subsysteme aufgehoben.

4.3.9 REMOVE-CATALOG-ENTRY

Subsystemdefinition logisch aus dem Subsystemkatalog löschen

Funktionsbeschreibung

Mit dieser Anweisung wird die Definition eines Subsystems, die im Subsystemkatalog hinterlegt ist, logisch gelöscht.

Die Anweisung wird zurückgewiesen, wenn kein aktueller (nicht leerer) Katalog, in dem die Definition hinterlegt ist, mit der Anweisung START-CATALOG-CREATION bzw. START-CATALOG-MODIFICATION eröffnet wurde.

Ist das angegebene Subsystem im bezeichneten Katalog nicht enthalten, wird die Anweisung ebenfalls zurückgewiesen. Wird das Subsystem aus dem Katalog entfernt, werden auch die disjunkten Beziehungen (die mit der Anweisung SEPARATE-ADDRESS-SPACE vereinbart wurden) gelöscht.

> ¡ Bei bestehenden Abhängigkeits- oder Ladebeziehungen kann das Entfernen eines Subsystems zu Fehlern bei der Konsistenzprüfung des Kataloges führen!

Format

REMOVE-CATALOG-ENTRY
SUBSYSTEM-NAME = <structured-name 1..8>(...)   <structured-name>(...)   **VERSION** = ***ALL** / <c-string 3..8> / <text 3..8>

Operandenbeschreibung

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name des Subsystems, dessen Definition aus dem Katalog gelöscht werden soll.

VERSION = *ALL / <c-string 3..8> / <text 3..8>
Spezifiziert die Version des Subsystems, dessen Definition aus dem Katalog gelöscht werden soll. Mit der Angabe VERSION=*ALL können die Definitionen aller Versionen des angegebenen Subsystems aus dem Katalog gelöscht werden.

4.3.10 SAVE-CATALOG

Subsystemkatalog als PAM-Datei abspeichern

Funktionsbeschreibung

Mit dieser Anweisung wird der durch zuvor eingegebene Anweisungen im Speicher aufgebaute Subsystemkatalog als PAM-Datei abgespeichert.

Der Dateiname wird entweder aus der letzten Anweisung START-CATALOG-CREATION bzw. der Anweisung START-CATALOG-MODIFICATION abgeleitet, oder es kann ein neuer, vollqualifizierter Dateiname angegeben werden.

SAVE-CATALOG wird mit Fehlermeldung abgewiesen, wenn:

- vorher keine der Anweisungen START-CATALOG-CREATION oder START-CATALOG-MODIFICATION ausgeführt wurde
- der im Speicher aufgebaute Katalog keine Definitionen enthält, also leer ist
- wenn bereits eine Datei mit dem bei CATALOG-NAME angegebenen Namen (ungleich *CURRENT) katalogisiert ist und REPLACE-OLD-FILE=*NO angegeben wurde

Zu beachten ist auch, dass dieser Anweisung ein Prüflauf auf den Katalog vorausgehen muss. Dieser Prüflauf kann mit der Anweisung CHECK-CATALOG eingeleitet werden. Für die Operanden DEPENDENCE-RELATION und LINK-RELATION muss dazu *YES angegeben werden.

Format

SAVE-CATALOG
CATALOG-NAME = ***CURR** ENT / <filename 1..51 without-gen-vers>(...) **REPLACE-OLD-FILE** = ***NO** / ***Y**ES ,**FORCED** = ***NO** / ***Y**ES / ***FOR-ADD-SUBSYSTEM**

Operandenbeschreibung

CATALOG-NAME =
Dateiname, unter dem der Katalog abgespeichert werden soll.

CATALOG-NAME = *CURRENT
Der Katalog wird unter dem Dateinamen abgespeichert, der in der letzten Anweisung START-CATALOG-CREATION oder START-CATALOG-MODIFICATION angegeben wurde.

CATALOG-NAME = <filename 1..51 without-gen-vers>(...)
Vollqualifizierter Dateiname, unter dem der Katalog gespeichert werden soll.

REPLACE-OLD-FILE= *NO / *YES
Existiert bereits eine katalogisierte Datei mit dem bei CATALOG-NAME angegebenen Namen, legt REPLACE-OLD-FILE fest, ob sie überschrieben werden darf (*YES) oder nicht (*NO).

FORCED =
Bestimmt, wie SSCM bei Fehlern in der Subsystemdefinition verfahren soll.

FORCED = *NO
Voreinstellung: bei Fehlern, die SSCM bei der Analyse der Subsystemdefinition erkennt, soll der Katalog nicht in einer PAM-Datei abgespeichert werden.
Fehler bei der Definition von Subsystemen könnten sich aus einer nicht-zyklenfreien Konstellation an Beziehungen (Adress-, Abhängigkeits- oder Externverweisbeziehungen) ergeben. Ein solcher Zyklus entsteht dann, wenn eine Beziehung („-->“) definiert wird, die nach mehreren Schritten wieder auf sich selbst verweist:

Subsystem A --> Subsystem B
Subsystem B --> Subsystem C
Subsystem C --> Subsystem A

FORCED = *YES
Trotz möglicher Fehler, die SSCM bei der Analyse der Subsystemdefinition erkennen könnte, soll der Katalog in einer PAM-Datei abgespeichert werden.
Die Verantwortung für einen so gespeicherten Katalog obliegt dem Anwender; das Verhalten von DSSM kann nicht gewährleistet werden.

FORCED = *FOR-ADD-SUBSYSTEM
Der angegebene Katalog wird trotz folgender Fehler gesichert:

- ein Subsystem weist Abhängigkeitsbeziehungen zu nicht definierten Subsystemen auf
- ein Subsystem soll sich mit einem nicht definierten Subsystem eine Holdertask teilen

Dieser Wert ist dann sinvoll, wenn ein Katalog erstellt werden soll, der lediglich neue Subsysteme enthält, die aber z.B. zu anderen, bereits im alten Katalog definierten Subsystemen Beziehungen haben. Zum bestehenden alten Katalog wird dieser neue mit dem Kommando ADD-SUBSYSTEM ...,TYPE=*NEW-SUBSYSTEMS hinzugefügt.

4.3.11 SAVE-SSD

Subsystemdefinition(en) abschließen

Funktionsbeschreibung

Die Anweisung SAVE-SSD schließt die Folge von Anweisungen zur Definition eines oder mehrerer Subsysteme ab, die mit den Anweisungen START-SSD-CREATION und SET-SUBSYSTEM-ATTRIBUTES eingeleitet wurde.
Die Anweisung veranlasst SSCM dazu, die Subsystemdefinition(en) in die in der Anweisung START-SSD-CREATION benannte ISAM-Datei abzuspeichern. Diese ISAM-Datei wird SSD-Object genannt.

SAVE-SSD wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-SSD-CREATION
- SET-SUBSYSTEM-ATTRIBUTES

SAVE-SSD wird mit einer Fehlermeldung abgebrochen, wenn das SSD-Object Subsysteme enthält, die

- mit CLOSE-CTRL-ROUTINE=*DYNAMIC definiert wurden, deren Holdertask aber nicht als Arbeitstask genutzt wird
- die verschiedene INSTALLATION-UNIT-Namen (installierte Liefergruppen) haben.

Format

SAVE-SSD

4.3.12 SEPARATE-ADDRESS-SPACE

Disjunkte Verteilung der Subsysteme im Klasse-5-Speicher steuern

Funktionsbeschreibung

Mit dieser Anweisung wird die disjunkte Verteilung der Subsysteme im Klasse-5-Speicher gesteuert. Mithilfe dieser Anweisung ist es möglich, unerwünschte Adressraumüberschneidungen von Subsystemen, die sich aus der Anweisung SET-SUBSYSTEM-ATTRIBUTES ergeben könnten, zu vermeiden.

Die Angabe ist für Subsysteme im Klasse-3- oder Klasse-4-Speicher (Operand MEMORY-CLASS=*SYSTEM-GLOBAL in der Anweisung SET-SUBSYSTEM-ATTRIBUTES) sowie für Subsysteme im Klasse-6-Speicher (MEMORY-CLASS=*LOCAL-UNPRIVILEGED) nicht relevant.

Subsystem im Klasse-6-Speicher werden nie parallel gebraucht, dürfen sich also damit im Adressraum überschneiden, und Subsysteme im Klasse-3- oder Klasse-4-Speicher überschneiden sich grundsätzlich nicht.

SEPARATE-ADDRESS-SPACE wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-SSD-CREATION
- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Format

SEPARATE-ADDRESS-SPACE
SUBSYSTEM-NAME = <structured-name 1..8> ,**FROM-SUBSYSTEMS** = list-poss(15): <structured-name 1..8>

Operandenbeschreibung

SUBSYSTEM-NAME = <structured-name 1..8>
Name des Subsystems, das sich nicht mit anderen Subsystemen überschneiden darf. Das Subsystem muss SSCM bekannt und bereits definiert sein.

FROM-SUBSYSTEMS = list-poss(15): <structured-name 1..8>
Liste von maximal 15 Subsystemen, mit denen sich das im Operanden SUBSYSTEM-NAME angeführte Subsystem nicht überschneiden darf.

Hinweise

Die Anweisung SEPARATE-ADDRESS-SPACE erzeugt immer mindestens zwei disjunkte Beziehungen, wie am nachfolgenden Beispiel zu sehen ist:

```
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=EINS,FROM-SUBSYSTEMS=(ZWEI,DREI)
```

Die Ausführung dieser Anweisung schafft folgende Beziehungen:

- der Adressraum des Subsystems EINS ist disjunkt zum Adressraum des Subsystems ZWEI
- der Adressraum des Subsystems EINS ist disjunkt zum Adressraum des Subsystems DREI
- der Adressraum des Subsystems ZWEI ist disjunkt zum Adressraum des Subsystems EINS

- der Adressraum des Subsystems DREI ist disjunkt zum Adressraum des Subsystems EINS

4.3.13 SET-SUBSYSTEM-ATTRIBUTES

Eigenschaften und Einsprungstellen eines Subsystems definieren

Funktionsbeschreibung

Mit dieser Anweisung können alle Eigenschaften und Einsprungstellen eines Subsystems definiert werden.

In welcher Weise die Definition eines neuen Subsystems aufgebaut wird, ist abhängig davon, welche Anweisung der Definition vorausging:

- Nach der Anweisung START-CATALOG-CREATION kann das Subsystem in den **Katalog** integriert werden.
- Nach der Anweisung START-SSD-CREATION kann das Subsystem in das **SSD-Object** integriert werden. In ein und demselben SSD-Object können jedoch nicht verschiedene Versionen des gleichen Subsystems angelegt werden.

Subsysteme, die mit dem Operanden
MEMORY-CLASS=*SYSTEM-GLOBAL ..., SUBSYSTEM-ACCESS=*SYSTEM definiert werden, sind privilegierte Subsysteme.

Bei der Definition ist ferner auf Folgendes zu achten:

- Der Name des Subsystems und seine Version dürfen noch nicht im Subsystemkatalog bekannt sein.
- Zwei verschiedene Versionen ein- und desselben Subsystems können nicht in einem SSD-Object definiert werden.
- Der Subsystem-Name CP ist für DSSM reserviert und darf nicht angegeben werden.
- Die Namen der Einsprungstellen dürfen nicht doppelt genannt werden.
- Der Name des zu definierenden Subsystems darf nicht in der Liste der Subsysteme enthalten sein, zu denen Adress- oder Abhängigkeitsbeziehungen bestehen.
- Innerhalb dieser Liste darf keine Doppelnennung eines Subsystems auftauchen.

SET-SUBSYSTEM-ATTRIBUTES wird abgewiesen, wenn vorher keine der folgenden Anweisungen ausgeführt wurde:

- START-SSD-CREATION
- START-CATALOG-CREATION
- START-CATALOG-MODIFICATION

Hinweis zur Syntax

Für die Namen der Einsprungstellen in den folgenden Operanden (im Format ist der Datentyp <name> angegeben) kann auch der spezielle Datentyp <symbol> verwendet werden, der im Handbuch „BLSSERV “ ausführlich beschrieben ist:

- LINK-ENTRY
- DYNAMIC-CHECK-ENTRY
- INIT-ROUTINE
- CLOSE-CTRL-ROUTINE
- DEINIT-ROUTINE
- INTERFACE-VERSION

- SUBSYSTEM-ENTRIES
- STOPCOM-ROUTINE

Format

SET-SUBSYSTEM-ATTRIBUTES

SUBSYSTEM-NAME = <structured-name 1..8>(...)

<structured-name 1..8>(...)

VERSION = <c-string 3..8> / <text 3..8>

,**INSTALLATION-UNIT** = ***NONE** / ***STD** / <text 1..30>

,**INSTALLATION-USERID** = ***NONE** / ***DEF**AULT**-USERID** / <name 1..8>

,**COPYRIGHT** = ***NONE** / <c-string 1..54>(...)

<c-string 1..54>(...)

YEAR = ***YEAR-1990** / <c-string 4..4>

,**LIB**RARY = ***STD** / ***CPLINK** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>

,**DEFAULT-NAME** = <filename 1..54 without-gen-vers>

,**SUBSYSTEM-LOAD-MODE** = ***ANY** / ***STD** / ***ADVANCED**

,**REP-FILE** = ***STD** / ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>

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

,**REP-FILE-MANDATORY** = ***NO** / ***Y**ES

,**MES**SAGE**-FILE** = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>

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

,**SUBSYSTEM-INFO-FILE** = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>

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

,**SYNTAX-F**ILE = ***NO** / ***INSTALLED**(...) / <filename 1..54 without-gen-vers>

***INSTALLED**(...)

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>

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

,**DYNAMIC-CHECK-ENTRY** = ***STD** / ***NO** / <text 1..8 without-sep>

,**CRE**ATION**-TIME** = ***AT-CRE** ATION **-REQ** UEST / ***AT-SUBSYS**TEM**-CALL**(...) / ***AT-DSSM-LOAD** /
***BEFORE-DSSM-LOAD** / ***MANDATORY-AT-STARTUP** / ***BEFORE-SYSTEM-READY** /
***AFTER-SYSTEM-READY**

***AT-SUBSYS**TEM**-CALL**(...)

ON-ACTION = ***STD** / ***ISL-CALL** / *ANY

,**INIT-ROUTINE** = ***NO** / <text 1..8 without-sep>

,**CLOSE-CTRL-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOPCOM-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**DEINIT-ROUTINE** = ***NO** / ***DYN**AMIC / <text 1..8 without-sep>

,**STOP-AT-SHUTDOWN** = ***NO** / ***Y**ES

,**INTERFACE-VERSION** = ***NO** / <text 1..8 without-sep>

,**SUBSYSTEM-HOLD** = ***ALLOW** ED / ***FORBIDDEN**

,**STATE-CHANGE-CMDS** = ***ALLOW** ED / ***FORBIDDEN** / ***BY-ADMINISTRATOR-ONLY**

,**FORCED-STATE-CHANGE** = ***FORBIDDEN** / ***ALLOW**ED

,**RESET** = ***FORBIDDEN** / ***ALLOW**ED

,**RESTART-REQUIRED** = ***NO** / ***Y**ES

,**VERSION-COEXISTENCE** = ***FORBIDDEN** / ***ALLOW**ED

,**VERSION-EXCHANGE** = ***FORBIDDEN** / ***ALLOW**ED

,**SUBSYSTEM-ENTRIES** = ***NONE** / list-poss(100): <text 1..8>(...) **/ *BY-PROGRAM(...)**

<text 1..8>(...)

MODE = ***LINK** / ***ISL**(...) / ***SVC**(...) / ***SYSTEM-EXIT**(...)

***ISL**(...)

FUNCTION-NUMBER = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SVC**(...)

NUMBER = <integer 0..255>

,**CALL-BY-SYSTEM-EXIT** = ***ALLOW** ED / ***FORBIDDEN**

,**FUNCTION-NUMBER** = ***NONE** / <integer 0..255>(...)

<integer 0..255>(...)

FUNCTION-VERSION = <integer 1..255>

***SYSTEM-EXIT**(...)

NUMBER = <integer 0..127>

,**CONNECTION-ACCESS** = ***ALL** / ***SYS**TEM / ***SIH**

,**CONNECTION-SCOPE** = ***TASK** / ***PROG**RAM / ***FREE** / ***CALL** / ***OPTIMAL**

, **FIRST-CONNECTION** = ***ALLOW** ED / ***FORBIDDEN**

***BY-PROGRAM(...)**

CONNECTION-SCOPE = ***TASK** / ***PROGRAM**

,**MEMORY-CLASS** = ***SYSTEM-GLOBAL** (...) / ***LOCAL-PRIVILEGED**(...) / ***LOCAL-UNPRIVILEGED**(...) /

***BY-SLICE**(...)

***SYSTEM-GLOBAL**(...)

SUBSYSTEM-ACCESS = ***LOW** / ***SYS**TEM / ***HIGH**

***LOCAL-PRIVILEGED**(...)

SIZE = <integer 1..32767>

***LOCAL-UNPRIVILEGED**(...)

SIZE = <integer 1..32767>

,**SUBSYSTEM-ACCESS** = ***LOW** / ***HIGH**

,**START-ADDRESS** = ***ANY** / <x-string 7..8>

***BY- SLICE**(...)

SIZE = <integer 1..32767>

,**LINK-ENTRY** = <text 1..8 without-sep>(...)

<text 1..8 without-sep>(...)

AUTOLINK = ***FORBIDDEN** / ***ALLOW**ED

,**REFERENCED-SUBSYSTEM** = ***NONE** / list-poss(15): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***LOW** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***HIGH** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**UNRESOLVED-EXTERNALS** = ***FORBIDDEN** / ***ALLOW**ED

,**CHECK-REFERENCE** = ***Y** ES / ***NO**

,**RELATED-SUBSYSTEM** = ***NONE** / list-poss(100): <structured-name 1..8>(...)

<structured-name 1..8>(...)

LOWEST-VERSION = ***LOW** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

,**HIGHEST-VERSION** = ***HIGH** EST **-EXIST** ING / <c-string 3..8> / <text 3..8>

Operandenbeschreibung

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Vereinbart Name und Version des zu definierenden Subsystems.

VERSION = <c-string 3..8> / <text 3..8>
Die Version des Subsystems ist im Format „[V][m]m.n[aso]“ zu vereinbaren, wobei die Textteile folgende Bedeutung besitzen:

mm: Hauptversion (numerisch)

n: Nachtragsversion (numerisch)

aso: Änderungsstand (a: Buchstabe, Freigabestand; so: numerisch, Korrekturstand)

INSTALLATION-UNIT =
Legt den Namen der installierten Liefergruppe fest. Für alle mit IMON zu installierenden Subsysteme muss ein Wert ungleich *NONE angegeben werden, ebenso, wenn bei den Operanden SUBSYSTEM-LIBRARY, REP-FILE, SUBSYSTEM-INFO-FILE, MESSAGE-FILE und

SYNTAX-FILE der Wert *INSTALLED(LOGICAL-ID=...) definiert wurde.
Die im Handbuch „IMON“ dargestellten Syntaxregeln sind bei der Festlegung des Namens zu beachten.

INSTALLATION-UNIT = *NONE
Voreinstellung: Es wird kein Name vergeben. Für alle mit IMON installierten Subsysteme ist diese Voreinstellung nicht erlaubt.

INSTALLATION-UNIT = *STD
Der beim Operanden SUBSYSTEM-NAME angegebene Name wird als Name der installierten Liefergruppe genutzt.

INSTALLATION-UNIT = <text 1..30>
Name der installierten Liefergruppe.

INSTALLATION-USERID =
Vereinbart eine Benutzerkennung, unter der die zuständige DSSM-Task die Nebenkomponenten des Subsystems (Rep-Datei, Objektmodulbibliothek, Meldungsdatei, Syntaxdatei und Subsystem-Informationsdatei) erwartet, falls diese Dateien noch keiner Benutzerkennung zugeordnet sind, d.h. der Dateiname ohne Benutzerkennung angegeben wurde.

INSTALLATION-USERID = *NONE
Die Dateien werden nicht unter einer bestimmten Benutzerkennung erwartet.

INSTALLATION-USERID = *DEFAULT-USERID
Die Dateien werden unter der System-Standardkennung erwartet (Prefix „$.“) bzw. unter der Benutzerkennung der aufrufenden Task, wenn es sich um ein lokales Subsystem handelt.

INSTALLATION-USERID = <name 1..8>
Benutzerkennung, unter der die Nebenkomponenten erwartet werden.
Gilt die Anweisung für ein SSD-Object, werden die Dateien nur dann unter der hier angegebenen Benutzerkennung erwartet, wenn in der Anweisung ADD-CATALOG-ENTRY (Übernahme von Subsystemdefinitionen aus dem SSD-Object in den Katalog) keine Benutzerkennung angegeben wurde. Die Angabe bei ADD-CATALOG-ENTRY hat Vorrang.

COPYRIGHT =
Vereinbart, ob und welche Copyright-Meldung beim Starten des Subsystems ausgegeben werden soll.

COPYRIGHT = *NONE
Voreinstellung: es soll keine Copyright-Meldung ausgegeben werden.

COPYRIGHT = <c-string 1..54>(...)
Text der Copyright-Meldung, die zusammen mit dem Datum der Erstellung beim Starten ausgegeben wird.

> **YEAR = *YEAR-1990 / <c-string 4..4>**
> Jahreszahl, die in der Copyright-Meldung als Datum der Erstellung erscheinen soll. Eine semantische Prüfung findet nicht statt.

LIBRARY =
Vereinbart den Namen der Programm- oder Bindemodulbibliothek (OML), aus der der Objectcode des Subsystems bei dessen Aktivierung geladen werden soll.

LIBRARY = *STD
Voreinstellung: der Objectcode wird beim Starten automatisch aus der Bibliothek SYSLNK.<subsysname>.<subsysvers#> geladen. Sie ist auf der Benutzerkennung abgelegt, unter der die Holdertask läuft; also auf der Benutzerkennung des Aufrufers bei lokalen Subsystemen und auf TSOS bei globalen Subsystemen.

Der Wert von „subsysvers#“ ist dreistellig und setzt sich aus den beim Operanden SUBSYSTEM-NAME=...(VERSION=...) angegebenen Teilen „mmn“ zusammen.

LIBRARY = *CPLINK
Das zu definierende Subsysteme ist mit dem Basissystem des BS2000 (CP=Control Program) verknüpft und muss bereits vor der Aktivierung von DSSM geladen sein.
Der Operand darf nur in Verbindung mit dem Operanden CREATION-TIME =*BEFORE-DSSM-LOAD verwendet werden.

LIBRARY = *INSTALLED(...)
Der Bibliotheksname muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

> **LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>**
> Logischer Name der Programm- oder Bindemodulbibliothek.
>
> **DEFAULT-NAME = <filename 1..54 without-gen-vers>**
> Neuer Bibliotheksname bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

LIBRARY = <filename 1..54 without-gen-vers>
Vollqualifizierter Dateiname der Bindemodulbibliothek, aus der der Objectcode für das Subsystem geladen werden soll.

SUBSYSTEM-LOAD-MODE =
Bestimmt den Lademodus des Subsystems (über die BLS-DSSM-Schnittstelle $PBBND1).

SUBSYSTEM-LOAD-MODE = *ANY
Voreinstellung: das BLS wird im STD-Run-Mode aufgerufen und lädt das Subsystem als Objektmodul. Tritt während des Ladens ein Fehler ein, wird BLS ein zweites Mal aufgerufen. Der Aufruf erfolgt im ADVANCED-Run-Mode und das Subsystem wird als Bindelademodul (LLM) geladen.
Dieser Operandenwert wird nur aus Kompatibilitätsgründen weiter unterstützt.

SUBSYSTEM-LOAD-MODE = *STD
Das BLS wird im STD-Run-Mode aufgerufen und lädt das Subsystem als Objektmodul.

SUBSYSTEM-LOAD-MODE = *ADVANCED
Das BLS wird im ADVANCED-Run-Mode aufgerufen und lädt das Subsystem als Bindelademodul (LLM).

REP-FILE =
Legt fest, ob System-Reps für das zu definierende Subsystem benötigt werden und in welcher Datei diese hinterlegt sind. Diese Korrekturanweisungen werden während der Aktivierung des Subsystems ausschließlich auf die in der Bindemodulbibliothek hinterlegten und geladenen Module angewandt, nicht auf andere Subsysteme oder BS2000 CP.
Eine Rep-Datei kann auch für Module eines nicht-privilegierten Subsystems vereinbart werden.

REP-FILE darf nicht zusammen mit LIBRARY=*CPLINK angegeben werden.

REP-FILE = *STD
Standardmäßig werden die System-Reps aus der Rep-Datei mit dem Namen

SYSREP.<subsysname>.<subsysvers#> geladen. Die Datei ist auf der Benutzerkennung abgelegt, unter der die Holdertask läuft; also auf der Benutzerkennung des Aufrufers bei lokalen Subsystemen und auf TSOS bei globalen Subsystemen.
Der Wert von „subsysvers#“ ist dreistellig und setzt sich aus den beim Operanden SUBSYSTEM-NAME=...(VERSION=...) angegebenen Teilen „mmn“ zusammen.

REP-FILE = *NO
Für das Subsystem soll keine Rep-Datei verarbeitet werden.

REP-FILE = *INSTALLED(...)
Der Name der Rep-Datei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logischer Name der Rep-Datei.

DEFAULT-NAME =
Name der Rep-Datei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Es wird ein neuer Name vergeben.

DEFAULT-NAME = *NONE
Es wird kein neuer Name vergeben.

REP-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Rep-Datei, aus der die Korrekturanweisungen gelesen werden sollen.

REP-FILE-MANDATORY =
Legt fest, ob eine mit dem Operanden REP-FILE deklarierte Rep-Datei beim Laden des Subsystems abgearbeitet werden muss oder nicht.

REP-FILE-MANDATORY = *NO
Voreinstellung: der Einsatz einer Rep-Datei ist nicht Pflicht, d.h. weder die Rep-Datei noch deren Einträge sollen beim Aktivieren des Subsystems geprüft werden. Sollte die Rep-Datei nicht zugreifbar oder einzelne Korrekturanweisungen fehlerhaft sein, wird das Subsystem auch in diesem Fall gestartet.

REP-FILE-MANDATORY = *YES
Sollte es bei der Bearbeitung der Rep-Datei zu folgenden Fehlern kommen, wird der Versuch, das Subsystem zu laden, abgebrochen:

- Die Rep-Datei ist nicht katalogisiert oder kann nicht gelesen werden.
- Die Prüfung der Korrekturanweisungen zeigt einen Fehler an.
- Der Name der Korrekturanweisungen ist fehlerhaft.
- Das DMS meldet einer Fehler beim Zugriff auf die NOREF-Datei (diese Datei wird während des Ladens eines Subsystems benutzt, um zu verhindern, dass ungültige System-Reps an der Bedienstation protokolliert werden).

MESSAGE-FILE =
Bestimmt, ob es eine subsystemspezifische Meldungsdatei gibt, die beim Laden des Subsystems automatisch aktiviert wird. Für Subsysteme, die mit dem Startzeitpunkt AT-DSSM-LOAD definiert werden, ist im Operanden RELATED-SUBSYSTEM eine Abhängigkeitsbeziehung zum Subsystem MIP zu vereinbaren.

MESSAGE-FILE = *NO
Voreinstellung: es soll keine Meldungsdatei aktiviert werden. Dieser Wert ist verpflichtend für alle Subsysteme, die mit dem Startzeitpunkt BEFORE-DSSM-LOAD definiert werden (siehe auch Operand CREATION-TIME), da zu diesem Zeitpunkt noch keine Meldungsdatei aktiviert werden kann.

MESSAGE-FILE = *INSTALLED(...)
Der Name der Meldungsdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logischer Name der Meldungsdatei.

DEFAULT-NAME =
Name der Meldungsdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Es wird ein neuer Name vergeben.

DEFAULT-NAME = *NONE
Es wird kein neuer Name vergeben.

MESSAGE-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Meldungsdatei. Diese wird beim Laden des Subsystems automatisch aktiviert, beim Entladen automatisch deaktiviert (START-/ STOP-SUBSYSTEM).

SUBSYSTEM-INFO-FILE =
Bestimmt, ob eine Subsysteminformationsdatei (SSINFO) vorhanden ist. In dieser Datei sind subsystemspezifische Daten (Nebenkomponenten und Konfigurationsdaten) enthalten, die nicht von DSSM zentral bearbeitet werden können.

SUBSYSTEM-INFO-FILE = *NO
Voreinstellung: eine Informationsdatei für das Subsystem ist nicht verfügbar.

SUBSYSTEM-INFO-FILE = *INSTALLED(...)
Der Name der Informationsdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

> **LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>**
> Logischer Name der Informationsdatei.
>
> **DEFAULT-NAME =**
> Name der Informationsdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.
>
> **DEFAULT-NAME = <filename 1..54 without-gen-vers>**
> Es wird ein neuer Name vergeben.
>
> **DEFAULT-NAME = *NONE**
> Es wird kein neuer Name vergeben.

SUBSYSTEM-INFO-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Informationsdatei. Der Name wird automatisch an die Aktivierungs- und Deaktivierungsroutinen (Operanden INIT-/DEINIT-/STOPCOM-/CLOSE-CTRL-ROUTINE) übergeben, wenn diese aufgerufen werden.

SYNTAX-FILE =
Bestimmt, ob eine Syntaxdatei mit dem Subsystem verknüpft ist, die beim Laden des Subsystems automatisch aktiviert wird. Für Subsysteme, die mit dem Startattribut MANDATORY-AT-STARTUP definiert werden, ist im Operanden RELATED-SUBSYSTEMS eine Abhängigkeitsbeziehung zum Subsystem SDF zu deklarieren.

SYNTAX-FILE = *NO
Voreinstellung: es soll keine Syntaxdatei aktiviert werden. Dieser Wert ist verpflichtend für alle Subsysteme, die mit dem Startzeitpunkt BEFORE-DSSM oder AT-DSSM-LOAD definiert werden (siehe auch Operand CREATION-TIME), da zu diesem Zeitpunkt noch keine Syntaxdatei aktiviert werden kann.

SYNTAX-FILE = *INSTALLED(...)
Der Name der Syntaxdatei muss durch den Aufruf von IMON-GPN (Verwaltung von Installationspfaden) bestimmt werden.
Wird eine der Nebenkomponenten mit einem logischen Namen angesprochen, müssen bei allen zu diesem Subsystem gehörenden Nebenkomponenten logische Namen angegeben werden. Wird ein logischer Name vergeben, muss beim Operanden INSTALLATION-UNIT ein Wert ungleich *NONE vergeben werden.

LOGICAL-ID = <filename 1..30 without-catid-userid-gen-vers>
Logischer Name der Syntaxdatei.

DEFAULT-NAME =
Name der Syntaxdatei bei Nichtverfügbarkeit von IMON-GPN oder wenn der logische Name unbekannt ist.

DEFAULT-NAME = <filename 1..54 without-gen-vers>
Es wird ein neuer Name vergeben.

DEFAULT-NAME = *NONE
Es wird kein neuer Name vergeben.

SYNTAX-FILE = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der Syntaxdatei, die beim Laden des Subsystems automatisch aktiviert werden soll.

DYNAMIC-CHECK-ENTRY =
Vereinbart, ob eine dynamische Identitätsprüfung des Subsystems vorgenommen werden soll. Zu diesem Zweck muss eine Einsprungstelle angegeben werden, an der sowohl der Subsystemname (acht Zeichen) als auch die Versionsnummer (vier bzw. sieben Zeichen) stehen muss. DSSM prüft, ob die bei der Definition vergebene Identifikation mit dem geladenen Subsystem übereinstimmt.

DYNAMIC-CHECK-ENTRY = *STD
Als Voreinstellung gilt, dass die bei dem Operanden LINK-ENTRY spezifizierte Einsprungstelle für die Identitätsprüfung herangezogen werden soll.

DYNAMIC-CHECK-ENTRY = *NO
Eine Überprüfung soll nicht stattfinden. Dieser Operandenwert darf allerdings für solche Subsysteme, die vor der Aktivierung von DSSM geladen sein sollen (CREATION-TIME=*BEFORE-DSSM-LOAD), nicht verwendet werden.

DYNAMIC-CHECK-ENTRY = <text 1..8 without-sep>
Name der Einsprungstelle, die für die Identitätsprüfung herangezogen werden soll.

CREATION-TIME =
Legt den Zeitpunkt fest, an dem die Aktivierung des Subsystems (CREATE-Routine) angestoßen wird.
Während der Systemeinleitung sind zwei Phasen zu unterscheiden, in denen DSSM nach Aufruf durch die Startup-Routine die Steuerung der Systemeinleitung übernimmt:

Phase 1: Der DSSM-Code wird geladen, die DSSM-Task generiert und gestartet.

Die Task reserviert Klasse-5-Speicher, liest den Subsystemkatalog ein und startet die Subsysteme, die mit den Startattributen BEFORE-DSSM-LOAD und AT-DSSM-LOAD definiert wurden.

Nach dem Laden dieser Subsysteme geht die Steuerung der Systemeinleitung an die Startup-Routine zurück.

Phase 2: Nach erneutem Aufruf werden alle Subsysteme geladen, die mit den Startattributen MANDATORY-AT-STARTUP, BEFORE-SYSTEM-READY und AFTER-SYSTEM-READY definiert wurden.

Bei den beiden erstgenannten wird das Laden der Subsysteme mit der Startup-Routine synchronisiert (d.h. das Laden muss abgeschlossen sein), beim letztgenannten wird das Laden asynchron angestoßen.

Die Steuerung der Systemeinleitung geht an die Startup-Routine zurück.

Sollen verschiedene Versionen eines Subsystems definiert werden, können die Startattribute, die für die Phasen 1 und 2 der Systemeinleitung vorgesehen sind, nur für eine dieser Versionen vergeben werden.

CREATION-TIME = *AT-CREATION-REQUEST
Voreinstellung: das Subsystem muss explizit mit dem Kommando START-SUBSYSTEM geladen werden.

CREATION-TIME = *AT-SUBSYSTEM-CALL(...)
Das Subsystem soll automatisch beim ersten SVC- oder ISL-Aufruf geladen werden. Dieser Operandenwert ist reserviert für Subsysteme, die über SVC oder ISL aufgerufen werden.
Sind mehrere Versionen eines Subsystems mit diesem Operandenwert definiert, muss für alle diese Versionen VERSION-COEXISTENCE=*ALLOWED angegeben werden sowie FUNCTION-NUMBER und FUNCTION-VERSION für ihre SVC- bzw. ISL-Einsprungstellen, die mit CONNECTION-ACCESS ungleich *SIH deklariert wurden.

Mindestens eines der angegebenen Subsysteme muss mit SUBSYSTEM-ENTRIES ..., MODE=*SVC/*ISL deklariert worden sein (übereinstimmend mit dem Wert des Operanden ON-ACTION).

ON-ACTION =
Bestimmt, wodurch das automatische Laden des Subsystems veranlasst wird.

ON-ACTION = *STD
Voreinstellung: das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden SVC-Einsprungstelle.

ON-ACTION = *ISL-CALL
Das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden ISL-Einsprungstelle.

ON-ACTION = *ANY
Das Laden beginnt beim Aufruf einer beliebigen, zum Subsystem gehörenden SVC- oder ISL-Einsprungstelle.

CREATION-TIME = *AT-DSSM-LOAD
Das Subsystem soll während der Systemeinleitung (Phase 1) unter der Kontrolle der DSSM-Task geladen werden. Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die ebenfalls mit diesem Startattribut oder dem Startattribut BEFORE-DSSM-LOAD definiert sind.
Die Dateien für dieses Subsystem müssen unter der Benutzerkennung TSOS auf dem Home-Pubset angelegt sein, da zum Startzeitpunkt weder der Benutzerkatalog zugreifbar, noch die IMPORT-PUBSET-Verarbeitung abgeschlossen ist.
Für diese Subsysteme ist die Angabe einer Syntaxdatei nicht zulässig.

CREATION-TIME = *BEFORE-DSSM-LOAD
Das Subsystem soll während der Systemeinleitung (Phase 1), aber nicht unter der Kontrolle der DSSM-Task geladen werden.
Solche Subsysteme sind mit dem Organisationsprogramm verknüpft und brauchen - bei der Aktivierung - nicht mit der DSSM-Task synchronisiert werden. Nach dem Laden des Subsystems läuft dieses allerdings wieder unter der Kontrolle von DSSM und kann aus Sicht des Anwenders wie andere Subsysteme gesteuert werden.
Anschluss- oder Abhängigkeitsbeziehungen zu Subsystemen, die mit einem anderen Startattribut definiert wurden, sind nicht möglich. Auch die Einbindung einer Meldungs- oder Syntaxdatei ist nicht zulässig. Alle Auftragseingänge (Operand SUBSYSTEM-ENTRIES) müssen deklariert sein, da DSSM die (privilegierte) Verbindung zu diesen Auftragseingängen herstellt. Es liegt in der vollen Verantwortung des Subsystem-Entwicklers, sicherzustellen, dass zu jedem Zeitpunkt mindestens eine Version dieses Subsystems verfügbar ist (z.B. PLAM). Der Name des Link-Kontextes für diese Subsysteme muss eindeutig sein, da DSSM auch eine Entlade-Anforderung erfüllen muss, selbst wenn DSSM den Subsystem-Code nicht geladen hat. Eine Einsprungstelle (Operand DYNAMIC-CHECK-ENTRY) muss angegeben sein.

CREATION-TIME = *BEFORE-SYSTEM-READY
Das Subsystem soll während der Systemeinleitung (Phase 2) geladen werden. Die Aktivierung wird synchron angestoßen; die Steuerung geht erst nach dem vollständigen Laden (oder nach Lade-Fehler) an die Startup-Routine zurück, die dann „SYSTEM READY“ melden kann.

Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD, AT-DSSM-LOAD oder MANDATORY-AT-STARTUP definiert wurden. Die Dateien für dieses Subsystem müssen auf dem Home-Pubset liegen. Wird ein nicht-privilegiertes Subsystem mit diesem Operandenwert deklariert, bekommt es implizit den Wert *AFTER-SYSTEM-READY zugewiesen. SSCM gibt eine Meldung aus.

CREATION-TIME = *MANDATORY-AT-STARTUP
Das Subsystem muss während der Systemeinleitung (Phase 2) geladen werden. Die Aktivierung wird - wie bei BEFORE-SYSTEM-READY - synchron angestoßen. Im Unterschied zum oben genannten muss das Laden des Subsystems allerdings **erfolgreich** abgeschlossen werden. Andernfalls geht eine Meldung an die Startup-Routine, dass ein verpflichtendes Subsystem nicht geladen werden konnte. Die Startup-Routine entscheidet in diesem Fall, ob die Verarbeitung fortgesetzt oder abgebrochen wird.
Das Subsystem muss privilegiert sein und darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD oder AT-DSSM-LOAD definiert wurden. Die Dateien für dieses Subsystem müssen auf dem Home-Pubset liegen.

CREATION-TIME = *AFTER-SYSTEM-READY
Das Laden des Subsystems wird während der Systemeinleitung (Phase 2) angestoßen. Die Durchführung dieses Auftrags wird nicht mit der Startup-Routine synchronisiert, die vor dem Abschluss des Ladens „SYSTEM READY“ melden kann.
Das Subsystem darf nur Adress- oder Abhängigkeitsbeziehungen zu Subsystemen aufweisen, die mit dem gleichen oder den Startattributen BEFORE-DSSM-LOAD, AT-DSSM-LOAD, MANDATORY-AT-STARTUP oder BEFORE-SYSTEM-READY definiert wurden. Die Dateien für dieses Subsystem müssen auf dem Home-Pubset liegen.

INIT-ROUTINE =
Legt fest, ob eine Initialisierungsroutine für das Subsystem durchlaufen werden soll, wenn es gestartet oder fortgesetzt wird. In diesem Fall muss der Name einer Einsprungstelle bekannt sein und DSSM delegiert die Initialisierung an die Holdertask des betreffenden Subsystems. Für alle Subsysteme mit dem Startattribut BEFORE-DSSM-LOAD wird die Angabe einer Einsprungstelle unbedingt empfohlen. Beim Laden des Subsystems (d.h. dem Durchlaufen der Initialisierungsroutine) erhält dieses dann Kenntnis davon, dass DSSM die Kontrolle über Anschluss und Abbau von Beziehungen übernehmen kann. Eine INIT-Routine muss mit RESTART-REQUIRED=*YES deklariert werden.

INIT-ROUTINE = *NO
Voreinstellung: es wird keine Initialisierungsroutine durchlaufen.

INIT-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der Initialisierungsroutine.

Der Initialisierungsroutine wird in der Holdertask die Steuerung übergeben, damit sich das Subsystem initialisieren kann. Dazu werden ihr übergeben:

- der Name und die Version des Subsystems, wie im SSCMCAT definiert
- der Name der SSINFO-Datei, falls dieser im Operanden SUBSYSTEM-INFO-FILE spezifiziert wurde
- die Adresse der beim Laden und Binden angegebenen Einsprungstelle (LINK-ENTRY)
- der vom Dynamischen Bindelader verwendete Binder-Kontext-Name

- der Name des Memory-Pools (für Subsysteme im Klasse-5- oder Klasse-6-Speicher), damit sich das Subsystem beim Nachladen eigener Selectable Units/Load Units darauf beziehen kann
- der Name der Meldungsdatei
- die Adresse des Operanden SUBSYSTEM-PARAMETER, falls im Kommando START-SUBSYSTEM ein String angegeben wird

Am Ende der Initialisierung wird eine Rückmeldung des Subsystems erwartet, ob die Initialisierung erfolgreich durchgeführt wurde und ob die Holdertask als Arbeitstask genutzt werden soll (wird in der Anweisung ASSIGN-HOLDER-TASK). Abhängig davon steht die Task dann weiter unter DSSM-Kontrolle oder unter Kontrolle des Subsystems.

CLOSE-CTRL-ROUTINE =
Legt fest, ob eine spezielle Routine für das Subsystem durchlaufen werden soll, wenn es angehalten oder deaktiviert wird.
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).

Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE. Beim Ansprung dieser Routine wird sichergestellt, dass noch Verbindung zum Subsystem besteht.
Existiert eine CLOSE-CTRL-Routine, tritt bei einem Versionswechsel während der BS2000-Session keine Unterbrechung auf. Es existiert zu jedem Zeitpunkt genau eine gültige Version (entweder die alte Version ist noch verfügbar oder die neue Version ist bereits verfügbar). Ohne eine solche Routine beinhaltet ein Versionswechsel immer eine Anschlussunterbrechung, während die STOPCOM-Routine der alten Version und die INIT-Routine der neuen Version ablaufen (siehe dazu auch "Austausch von Subsystemversionen").

CLOSE-CTRL-ROUTINE = *NO
Voreinstellung: im betreffenden Subsystem ist keine Routine verankert, die beim Deaktivieren oder Anhalten des Subsystems durchlaufen werden soll.

CLOSE-CTRL-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse oder FITC. Die notwendigen Parameter werden vom Subsystem am Ende der INIT-Routine dynamisch an die CLOSE-CTRL-Routine übergeben und DSSM über die Identifikation der Börse bzw. FITC-Ports in Kenntnis gesetzt.
Voraussetzung für die Nutzung der CLOSE-CTRL-Routine sind die Angabe einer INIT-Routine (Operand INIT-ROUTINE) und einer STOPCOM-Routine (Operand STOPCOM-ROUTINE= *NO/*DYNAMIC).

Beim Aufruf der CLOSE-CTRL-Routine muss die Holdertask des Subsystems Arbeitstask sein (Anweisung ASSIGN-HOLDER-TASK).

CLOSE-CTRL-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

STOPCOM-ROUTINE =
Legt fest, ob eine Routine in das Subsystem eingebunden ist, die das aktive Beenden der Aufträge durchführen kann.
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).
Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE.
Beim Ansprung dieser Routine ist sichergestellt, dass aufrufende Tasks nicht mehr an das Subsystem angeschlossen werden. Tasks, die noch in Aufruf-Beziehung zum Subsystem stehen, bleiben davon unberührt.
Wird CLOSE-CTRL-ROUTINE=*DYNAMIC vereinbart, muss der Operand STOPCOM-ROUTINE=*NO/*DYNAMIC angegeben werden.

STOPCOM-ROUTINE = *NO
Voreinstellung: im betreffenden Subsystem ist keine Routine verankert.

STOPCOM-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse oder FITC. Die notwendigen Parameter werden vom Subsystem am Ende der CLOSE-CTRL-Routine oder (wenn eine solche nicht vorhanden ist) am Ende der INIT-Routine dynamisch an die STOPCOM-Routine übergeben. DSSM wird über die Identifikation der Börse bzw. FITC-Ports in Kenntnis gesetzt. Voraussetzung für die Nutzung der STOPCOM-Routine ist die Angabe einer INIT-Routine (Operand INIT-ROUTINE).

Beim Aufruf der STOPCOM-Routine muss die Holdertask des Subsystems als Arbeitstask genutzt werden (Anweisung ASSIGN-HOLDER-TASK).

STOPCOM-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

DEINIT-ROUTINE =
Legt fest, ob eine Routine in das Subsystem eingebunden ist, die die Deinitialisierung des Subsystems durchführen kann. Diese Deinitialisierungsroutine realisiert die Rückgabe der vom Subsystem angeforderten Betriebsmittel (Speicher, Dateien, Geräte).
Wird ein Subsystem (mit STOP-SUBSYSTEM oder HOLD-SUBSYSTEM) deaktiviert, so erhält es von DSSM in der Holdertask an der bezeichneten Einsprungstelle die Kontrolle oder es wird (bei *DYNAMIC) über Börsen- bzw. FITC-Linkage benachrichtigt (gesteuert von Rückmeldungen bei der Initialisierung).
Wird ein Subsystem mit einer INIT- und einer CLOSE-CTRL-Routine definiert, muss eine

DEINIT-Routine - mit dem gleichen Operandenwert wie die CLOSE-CTRL-Routine - angegeben werden.
Die übergebenen Parameter sind die gleichen wie bei INIT-ROUTINE. Beim Ansprung dieser Routine ist sichergestellt, dass aufrufende Tasks nicht mehr an das Subsystem angeschlossen werden und alle vorhandenen Aufruf-Beziehungen gelöst werden.

DEINIT-ROUTINE = *NO
Voreinstellung: im betreffenden Subsystem ist keine Deinitialisierungsroutine verankert, die die Rückgabe der Betriebsmittel veranlasst.

DEINIT-ROUTINE = *DYNAMIC
Der Aufruf dieser Routine erfolgt über Börse bzw. FITC. Die notwendigen Parameter werden vom Subsystem am Ende der STOPCOM-Routine oder (wenn eine solche nicht vorhanden ist) am Ende der CLOSE-CTRL-Routine oder am Ende der INIT-Routine, wenn weder eine STOPCOM- noch eine CLOSE-CTRL-Routine eingebunden ist, dynamisch an die

DEINIT-Routine übergeben. DSSM wird über die Identifikation der Börse bzw. des FITC-Ports in Kenntnis gesetzt. Voraussetzung für die Nutzung der DEINIT-Routine ist die Angabe einer INIT-Routine (Operand INIT-ROUTINE). Beim Aufruf der DEINIT-Routine muss die Holdertask des Subsystems als Arbeitstask genutzt werden (Anweisung ASSIGN-HOLDER-TASK).

DEINIT-ROUTINE = <text 1..8 without-sep>
Name der Einsprungstelle der betreffenden Subsystemroutine.

STOP-AT-SHUTDOWN =
Legt fest, ob das Subsystem nach Beendigung der Benutzertasks bei Shutdown automatisch entladen werden soll.

STOP-AT-SHUTDOWN = *NO
Voreinstellung: das Subsystem wird nicht automatisch entladen.
Die Angabe sollte nicht für Subsysteme verwendet werden, die Adressbeziehungen zu anderen Subsystemen besitzen, die mit STOP-AT-SHUTDOWN=*YES definiert sind.

STOP-AT-SHUTDOWN = *YES
Das Subsystem wird bei Shutdown automatisch entladen.
Diese Angabe wird ignoriert, wenn keine STOPCOM-, DEINIT- oder CLOSE-CTRL-Routine angegeben wird.

INTERFACE-VERSION =
Bezeichnet die Einsprungstelle, über die DSSM auf diejenige Schnittstellenversion zugreifen kann, die für den Aufruf der INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routinen benutzt werden soll.
Dieser Operand ist Pflicht für Subsysteme, für die eine Einsprungstelle angegeben wurde (INIT-, CLOSE-CTRL-, STOPCOM-, DEINIT-Routine).

INTERFACE-VERSION = *NO
Voreinstellung: das Subsystem ruft keine INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routine auf.

INTERFACE-VERSION = <text 1..8 without-sep>
Name der Einsprungstelle. Die Einsprungstelle verweist auf den Standardheader, in dem u.a. auch die Schnittstellenversion hinterlegt ist. Der Standardheader wird durch den Aufruf des Makros $ESMINT(I) mit MF=I/L generiert.
Dieser Operand ist Pflicht für Subsysteme, für die eine INIT-, DEINIT-, STOPCOM- oder CLOSE-CTRL-Routine definiert wurde.

SUBSYSTEM-HOLD =
Legt fest, ob das geladene Subsystem angehalten oder entladen werden kann.

SUBSYSTEM-HOLD = *ALLOWED
Voreinstellung: das geladene Subsystem kann angehalten und entladen werden.
Die Kommandos HOLD-SUBSYSTEM und STOP-SUBSYSTEM sind für dieses Subsystem zulässig.

SUBSYSTEM-HOLD = *FORBIDDEN
Die Kommandos HOLD-SUBSYSTEM und STOP-SUBSYSTEM sind für dieses Subsystem nicht zulässig; das Subsystem wird - entsprechend den Angaben im Operanden STOP-AT-SHUTDOWN - bei Shutdown entladen.

Wird das Subsystem durch Austausch mit einem anderen Subsystem entladen, so erfolgt der Austausch unterbrechungsfrei.

STATE-CHANGE-CMDS =
Legt fest, ob die DSSM-Kommandos zur Steuerung des Subsystems im laufenden Betrieb (START-SUBSYSTEM, STOP-SUBSYSTEM, HOLD-SUBSYSTEM und RESUME-SUBSYSTEM) verwendet werden dürfen.

Wird von einer Version eines Subsystems in eine andere gewechselt, wird der bei STATE-CHANGE-CMDS angegebene Wert für die auszuwechselnde Version nicht berücksichtigt.

STATE-CHANGE-CMDS = *ALLOWED
Voreinstellung: Die Kommandos dürfen an der Bedienstation und unter der privilegierten Benutzerkennung (Benutzerkennung, die mit dem Systemprivileg SUBSYSTEM-MANAGEMENT ausgestattet ist) verwendet werden.

STATE-CHANGE-CMDS = *FORBIDDEN
Die Kommandos dürfen generell nicht - weder an der Bedienstation noch unter der privilegierten Benutzerkennung - verwendet werden.

STATE-CHANGE-CMDS = *BY-ADMINISTRATOR-ONLY
Die Kommandos dürfen nur unter der privilegierten Benutzerkennung verwendet werden; für den Operator an der Bedienstation sind die Kommandos gesperrt.

FORCED-STATE-CHANGE =
Legt fest, ob innerhalb der Kommandos STOP-SUBSYSTEM und HOLD-SUBSYSTEM die Verwendung des Operanden FORCED=*YES zulässig ist. Mit dieser Funktion kann das unbedingte Deaktivieren des Subsystems erzwungen werden.

FORCED-STATE-CHANGE = *FORBIDDEN
Voreinstellung: das Deaktivieren des Subsystems kann nicht erzwungen werden. DSSM weist die Verwendung des Operanden FORCED in den entsprechenden Kommandos mit einer Fehlermeldung zurück.

FORCED-STATE-CHANGE = *ALLOWED
Die Verwendung des Operanden FORCED=*YES für dieses Subsystem ist zulässig. Dieser Operandenwert darf nicht zusammen mit SUBSYSTEM-HOLD=*FORBIDDEN angegeben werden.

RESET =
Legt fest, ob innerhalb der Kommandos START-SUBSYSTEM und RESUME-SUBSYSTEM die Verwendung des Operanden RESET=*YES zulässig ist. Mit dieser Funktion kann das unbedingte Laden bzw. Fortsetzen des Subsystems erzwungen werden, auch wenn sich das Subsystem im Zustand IN-DELETE bzw. IN-HOLD befindet.

RESET = *FORBIDDEN
Voreinstellung: das Aktivieren des Subsystems kann nicht erzwungen werden. DSSM weist die Verwendung des Operanden RESET in den entsprechenden Kommandos mit einer Fehlermeldung zurück.

RESET = *ALLOWED
Die Verwendung des Operanden RESET=*YES für dieses Subsystem ist zulässig.

Dieser Operandenwert darf nicht mit SUBSYSTEM-HOLD=*FORBIDDEN angegeben werden.

RESTART-REQUIRED =
Legt fest, ob bei abnormaler Beendigung der Holdertask die Initialisierungsroutine für das Subsystem durchlaufen werden soll.

RESTART-REQUIRED = *NO
Voreinstellung: die Initialisierungsroutine wird für einen Wiederanlauf des Subsystems nicht benutzt.

RESTART-REQUIRED = *YES
Die Initialisierungsroutine soll bei abnormaler Beendigung der Holdertask benutzt werden. Voraussetzung ist, dass die Durchführung dieser Routine im Operanden INIT-ROUTINE vorgesehen ist.

VERSION-COEXISTENCE =
Vereinbart, ob mehr als eine Version des gleichen Subsystems gleichzeitig aktiv sein darf.

VERSION-COEXISTENCE = *FORBIDDEN
Voreinstellung: Die aktuelle Version des Subsystems kann nicht gleichzeitig mit einer anderen Version des gleichen Subsystems koexistieren.

VERSION-COEXISTENCE = *ALLOWED
Die aktuelle Version des Subsystems kann gleichzeitig mit einer anderen Version des gleichen Subsystems koexistieren (Coexistence-Modus).
Bei der Definition des Auftragseingangs (Operand SUBSYSTEM-ENTRIES) darf keine indirekte Verbindung über System-Exit-Routinen gewählt werden. Sind verschiedene Versionen des gleichen Subsystems geladen, die mit dem gleichen Auftragseingang definiert wurden, wird der Anschluss immer an die höchste geladene Version realisiert. Greifen koexistente Subsysteme auf koexistente Syntaxdateien zu, müssen diese im SSD-Object deklariert sein und können nicht von SDF verwaltet werden.
Bei Anschlüssen über SVC und ISL ist jedoch eine Versionsauswahl über die Operanden FUNCTION-NUMBER und FUNCTION-VERSION möglich.

VERSION-EXCHANGE =
Vereinbart, ob das Laden eines Subsystems im Exchange-Modus erlaubt ist.
Der Exchange-Modus erlaubt die temporäre Koexistenz zweier Versionen des gleichen Subsystems. Wird die Version B eines Subsystems geladen, wenn bereits Version A des Subsystems aktiv ist, werden alle neuen Aufrufer an die Version B angeschlossen. Die Aufträge, die an die Version A angeschlossen sind, werden noch bearbeitet. Nach Bearbeitung aller Aufträge an Version A wird diese automatisch beendet.
Bei der Definition ist darauf zu achten, dass die zu ersetzende, „alte“ Version nicht abhängig von der ersetzenden, „neuen“ Version sein darf.

VERSION-EXCHANGE = *FORBIDDEN
Voreinstellung: Die aktuelle Version des Subsystems darf nicht ausgetauscht werden.

VERSION-EXCHANGE = *ALLOWED
Der Exchange-Modus, der die temporäre Koexistenz zweier Subsysteme erlaubt, ist für die aktuelle Subsystemversion zulässig.

SUBSYSTEM-ENTRIES =
Vereinbart die Einsprungstellen (Auftragseingänge), die mit dem Subsystem verbunden sind. Es können bis zu 100 Auftragseingänge vereinbart werden. Werden mehr als 100 Auftragseingänge für ein Subsystem gewünscht, können diese mit den Anweisungen MODIFY-SUBSYSTEM-ATTRIBUTES (für ein Subsystem im Katalog) bzw. ADD-SUBSYSTEM-ENTRIES (für ein Subsystem in einem SSD-Object) definiert werden.

SUBSYSTEM-ENTRIES = *NONE
Voreinstellung: es werden keine Einsprungstellen vereinbart.

SUBSYSTEM-ENTRIES = list-poss(100): <text 1..8>
Vereinbart durch Angabe des Namens der Einsprungstelle maximal 100 Auftragseingänge, deren Typ jeweils in den Unterstrukturen definiert werden muss.

MODE =
Legt den Typ eines vereinbarten Auftragseingangs für das Subsystem fest.

MODE = *LINK
Voreinstellung: der Auftragseingang kann nicht über indirektes Linkage erreicht werden, sondern nur über eine CONNECT-Beziehung mittels eines externen Binder-Symbols.
Bei verschiedenen Versionen des gleichen Subsystems, die das gleiche externe Binder-Symbol nutzen, stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.

MODE = *ISL(...)
Der Auftragseingang wird durch indirekte Verbindung über System Procedure Linkage (nur für privilegierte Subsysteme) erreicht. Wird zusätzlich noch eine Funktions- und Versionsnummer der ISL-Einsprungstelle spezifiziert, darf sich die Kombination aus Name der Einsprungstelle, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED) decken.

Bei ungleichen Subsystemen, deren Auftragseingang über die selbe ISL-Einsprungstelle erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die die selbe ISL-Einsprungstelle nutzen (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die die selbe ISL-Einsprungstelle nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist,ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standardheader der Parameterliste des Aufrufers hinterlegt ist.

Der Wert *ALL des Operanden CONNECTION-ACCESS ist für ISL-Einsprungstellen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der ISL-Einsprungstelle angesprochen werden soll, da die gleiche ISL-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Voreinstellung: es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der ISL-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten ISL-Funktionsnummer.

MODE = *SVC(...)
Der Auftragseingang wird durch indirekte Verbindung über Supervisor Call (SVC) erreicht.
Wird zusätzlich noch eine Funktions- und Versionsnummer der SVC-Einsprungstelle spezifiziert, darf sich die Kombination aus SVC-Nummer, Funktions- und Versionsnummer nicht mit einer anderen Kombination für die verschiedenen Subsysteme eines Kataloges oder die verschiedenen Versionen des gleichen Subsystems (bei Angabe von VERSION-COEXISTENCE=*ALLOWED) decken.
Bei ungleichen Subsystemen, deren Auftragseingang über den selben SVC erreicht werden soll, muss zur eindeutigen Unterscheidung die Funktions- und Versionsnummer angegeben werden.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen - (ohne Angaben zur Funktions- oder Versionsnummer), stellt DSSM automatisch den Anschluss an die höchste geladene Version des Subsystems her.
Bei verschiedenen Versionen des gleichen Subsystems, die den selben SVC nutzen und deren Funktions- und Versionsnummer ungleich *NONE ist, ist die Auswahl, an welche Version der Anschluss erfolgen soll, von der Funktions- und Versionsnummer abhängig, die im Standardheader der Parameterliste des Aufrufers hinterlegt ist.
Bei Angabe dieses Operandenwertes sollte der Operand CONNECTION-ACCESS vorzugsweise mit dem Wert *SYSTEM statt *ALL belegt werden.

NUMBER = <integer 0..255>
Nummer des SVCs, über den der Auftragseingang erreicht wird. In Verbindung mit CONNECTION-ACCESS=*ALL ist die Verwendung einer SVC-Nummer größer 191 nicht zulässig.

CALL-BY-SYSTEM-EXIT =
Legt fest, ob die angegebene SVC-Nummer von System-Exit-Routinen aus aufgerufen werden darf.

CALL-BY-SYSTEM-EXIT = *ALLOWED
Voreinstellung: der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen zulässig.

CALL-BY-SYSTEM-EXIT = *FORBIDDEN
Der Aufruf der angegebenen SVC-Nummer ist für System-Exit-Routinen nicht zulässig.

FUNCTION-NUMBER =
Vereinbart, ob eine bestimmte Funktions- und Versionsnummer der SVC-Einsprungstelle angesprochen werden soll, da die gleiche SVC-Einsprungstelle von verschiedenen Funktionen genutzt werden kann.

FUNCTION-NUMBER = *NONE
Voreinstellung: es soll keine bestimmte Funktions- oder Versionsnummer angesprochen werden.

FUNCTION-NUMBER = <integer 0..255>(...)
Nummer der SVC-Einsprungstelle. Die Version ist in der folgenden Unterstruktur zu benennen.

FUNCTION-VERSION = <integer 1..255>
Version der spezifizierten SVC-Funktionsnummer.

MODE = SYSTEM-EXIT(...)
Der Auftragseingang wird durch indirekte Verbindung über System-Exit-Routinen erreicht.
In Verbindung mit CONNECTION-ACCESS=*ALL ist die Verwendung dieses Operanden nicht zulässig.

NUMBER = <integer 0..127>
Nummer der System-Exit-Routine.

CONNECTION-ACCESS =
Vereinbart die Zugriffsberechtigung (Privilegierung) für das Subsystem.

CONNECTION-ACCESS = *ALL
Voreinstellung: privilegierte und nicht-privilegierte Programmläufe sind zugriffsberechtigt. In Verbindung mit MODE=*SYSTEM-EXIT/*ISL/*SVC (mit einer SVC-Nummer größer 191) ist die Verwendung dieses Operandenwertes nicht zulässig.

CONNECTION-ACCESS = *SYSTEM
Nur privilegierte Programmläufe sind zugriffsberechtigt.

CONNECTION-ACCESS = *SIH
Nur Tasks, die im Funktionszustand SIH laufen, sind zugriffsberechtigt.

Das aufgerufene Subsystem läuft ebenfalls im Funktionszustand SIH, d.h. es ist nicht unterbrechbar.

Die Angabe dieses Operandenwertes ist nur für Subsysteme zulässig, deren Auftragseingang definiert wird über

- System Procedure Linkage (MODE=*ISL(FUNCTION-NUMBER=*NONE))
- CONNECTION-SCOPE=*OPTIMAL
- MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=*SYSTEM)

CONNECTION-SCOPE =
Bezeichnet das Ereignis, das die automatische Auflösung des Anschlusses an den angegebenen Subsystem-Auftragseingang hervorruft.

CONNECTION-SCOPE = *TASK
Voreinstellung: der Anschluss wird bei Taskbeendigung aufgehoben.

CONNECTION-SCOPE = *PROGRAM
Der Anschluss wird spätestens bei Programmbeendigung aufgehoben. Zusammen mit MEMORY-CLASS=*LOCAL-UNPRIVILEGED darf nur CONNECTION-SCOPE=*PROGRAM angegeben werden.
Die Angabe dieses Operandenwertes wird für Subsysteme empfohlen, die mit SUBSYSTEM-ACCESS=*LOW/*HIGH oder MEMORY-CLASS=*BY-SLICE deklariert wurden.

CONNECTION-SCOPE = *FREE
DSSM soll keine Kontrolle von Anschlüssen zu diesen Auftragseingängen durchführen. Der Anschluss wird - außer bei einer expliziten Anforderung - nicht automatisch aufgelöst.
Um Probleme oder mögliche Fehler beim Entladen des Subsystems zu vermeiden, müssen die Anschlüsse vom Subsystem selbst verwaltet werden.

CONNECTION-SCOPE = *CALL
Nach Rückkehr aus diesem Auftragseingang führt DSSM automatisch die Auflösung der Anschlüsse durch. Dieser Operandenwert steht nur für Subsysteme zur Verfügung, deren Auftragseingang über System Procedure Linkage (ISL) oder Supervisor Call (SVC) definiert wird.

CONNECTION-SCOPE = *OPTIMAL
Das Subsystem wird deaktiviert bzw. angehalten, wenn keine Task mehr Anschluss zu diesem Auftragseingang hat.
Eine Routine, deren Einsprungstelle mit *OPTIMAL definiert wird, muss mit RETURN beendet werden.
Wird ein Auftragseingang eines Subsystems mit CONNECTION-SCOPE=*OPTIMAL definiert, sollten alle seine Auftragseingänge im Subsystemkatalog mit MODE !=*LINK definiert werden.

Während ein Subsystem deaktiviert oder angehalten wird, wird kein Aufruf des Subsystems mit CONNECTION-SCOPE=*OPTIMAL akzeptiert.

FIRST-CONNECTION =
Bestimmt, ob der Erst-Anschluss der Task an den angegebenen Auftragseingang des Subsystems erlaubt ist. Mindestens ein Auftragseingang eines Subsystems muss mit FIRST-CONNECTION=*ALLOWED definiert werden.

FIRST-CONNECTION = *ALLOWED
Voreinstellung: Der Erst-Anschluss an den angegebenen Auftragseingang ist erlaubt.

FIRST-CONNECTION = *FORBIDDEN
Der Anschluss an den angegebenen Auftragseingang über SVC oder ISL ist nicht erlaubt, wenn die Task nicht bereits an einen anderen Auftragseingang des Subsystems angeschlossen ist.
Die Angabe dieses Operandenwertes ist für Auftragseingänge, die mit MODE=*LINK/*SYSTEM-EXIT oder CONNECTION-ACCESS=*SIH definiert wurden, nicht erlaubt.

SUBSYSTEM-ENTRIES = *BY-PROGRAM(...)
Die Einsprungstellen des angegebenen Subsystems werden nicht statisch aus dem Katalog versorgt, sondern dynamisch zum Ladezeitpunkt aus dem BLS-Namensverzeichnis. Voraussetzung für diese Funktionalität ist der Einsatz von BLSSERV ab Version 2.1, das den Einsatz von Extended External Names (EEN-Namen) als Einsprungstellen für DSSM-Subsysteme unterstützt.

Der Wert kann nicht zusammen angegeben werden mit dem Operanden SUBSYSTEM-ACCESS=*SYSTEM.

Wenn die Einsprungstellen mit diesem Wert definiert sind, werden die Operanden MODE,

CONNECTION-ACCESS und FIRST-CONNECTION mit den Standardwerten versorgt.

> ¡ Der Wert *BY-PROGRAM ist nicht nur auf EEN-Namen beschränkt. Er kann auch für Subsysteme angegeben werden, die eine Vielzahl von Einsprungstellen aufweisen.

CONNECTION-SCOPE = *TASK / *PROGRAM
Der Anschluss wird bei Task- bzw. Programmbeendigung aufgehoben.

MEMORY-CLASS =
Vereinbart den subsystemspezifischen Adressraum, in den das Subsystem geladen werden soll. Mit diesem Operanden kann die Systembetreuung die Adressraumvorgaben, die für das jeweilige Subsystem gelten, den speziellen Anforderungen der Installation anpassen.

MEMORY-CLASS = *SYSTEM-GLOBAL(...)
Voreinstellung: das Subsystem wird in den Klasse-3- oder Klasse-4-Speicher geladen. Residente CSECTs erhalten Klasse-3-Speicher, alle anderen erhalten seitenwechselbaren Klasse-4-Speicher.

SUBSYSTEM-ACCESS =
Bezeichnet die Zugriffsrechte (Privilegierung) und die Lage des angeforderten Speichers.

SUBSYSTEM-ACCESS = *LOW
Voreinstellung: Es wird nicht-privilegierter Klasse-3- oder Klasse-4-Speicher unterhalb der 16-MByte-Grenze zugewiesen.

SUBSYSTEM-ACCESS = *SYSTEM
Subsysteme, die mit diesem Operandenwert deklariert werden, sind privilegierte Subsysteme, denen privilegierter Adressraum oberhalb der 16-MByte-Grenze zugewiesen wird.
Die Angabe dieses Operandenwertes ist für die Subsysteme verpflichtend, deren Auftragseingang über SVC (MODE=*SVC) erfolgt, oder für die eine INIT-, STOPCOM-, DEINIT- oder CLOSE-CTRL-Routine vereinbart wird.
Der Wert kann nicht zusammen mit einer Einsprungstelle angegeben werden, die zusammen mit CONNECTION-ACCESS=*ALL oder MODE=*LINK angegeben wurde.
Der Wert kann nicht zusammen angegeben werden mit dem Operanden SUBSYSTEM-ENTRIES=*BY-PROGRAM.

SUBSYSTEM-ACCESS = *HIGH
Es wird nicht-privilegierter Adressraum bis zu 2 GByte zugewiesen.

MEMORY-CLASS = *LOCAL-PRIVILEGED(...)
Das Subsystem erhält einen Memory Pool im nicht-privilegierten Klasse-5-Speicher, der unterhalb der 16-MByte-Grenze angelegt wird.
Die Angabe ist auf nicht-privilegierte Subsysteme zugeschnitten, die relativ viel Adressraum (+/- 1 MByte) beanspruchen und unterhalb der 16-MByte-Grenze anzulegen sind. Die Subsysteme werden in Memory Pools an der gleichen Adresse geladen, um mit dem knapp bemessenen Adressraum unterhalb 16 MByte zu haushalten. Obwohl solche Subsysteme parallel im gleichen Adressraum geladen werden, können sie nicht simultan von einer Task genutzt werden (siehe auch Anweisung SEPARATE-ADDRESS-SPACE).
Das Subsystem darf keine residenten CSECTs enthalten, da ansonsten ein späteres Aktivieren abgebrochen wird.

SIZE = <integer 1..32767>
Größe des benötigten Adressraums (in 4KByte-Seiten) für den Memory Pool im Klasse-5-Speicher. Der Wert ist mindestens so groß zu wählen, dass das Subsystem und evtl. von diesem nachgeladene Selectable Units /Load Units in vollem Umfang geladen werden können. Die obere Grenze ist generierungsabhängig.

MEMORY-CLASS = *LOCAL-UNPRIVILEGED(...)
Das Subsystem erhält einen Memory Pool im nicht-privilegierten Klasse-6-Speicher. Die Angabe ist für Subsysteme reserviert, die wie ein Programm ausführbar sind. Analog dazu muss deren Zugriffsberechtigung (Privilegierung) im Operanden CONNECTION-ACCESS mit dem Wert *ALL definiert werden.

Dieser Operandenwert darf nicht zusammen mit einer Einsprungstelle angegeben werden, die mit CONNECTION-ACCESS=*SYSTEM definiert wurde.

Das Subsystem darf keine residenten CSECTs enthalten, da ansonsten ein späteres Aktivieren abgebrochen wird. Wird dieser Operandenwert angegeben, ist zur Auflösung des Anschlusses an den angegebenen Subsystem-Auftragseingang nur CONNECTION-SCOPE=*PROGRAM erlaubt.

SIZE = <integer 1..32767>
Größe des benötigten Adressraums (in 4KByte-Seiten) für den Memory Pool im Klasse-6-Speicher. Der Wert ist mindestens so groß zu wählen, dass das Subsystem und evtl. von diesem nachgeladene Selectable Units /Load Units in vollem Umfang geladen werden können. Die obere Grenze ist generierungsabhängig.

SUBSYSTEM-ACCESS =
Bezeichnet die Lage des angeforderten Speicherplatzes.

SUBSYSTEM-ACCESS = *LOW
Voreinstellung: Es wird nicht-privilegierter Klasse-6-Speicher unterhalb der 16-MByte-Grenze zugewiesen.
Da diese Angabe auf Subsysteme zugeschnitten ist, die wie Programme ausführbar sind, ist die zusätzliche Angabe CONNECTION-SCOPE=*PROGRAM anzuraten.

SUBSYSTEM-ACCESS = *HIGH
Es wird nicht-privilegierter Klasse-6-Speicher bis zu 2 GByte zugewiesen.
Da diese Angabe auf Subsysteme zugeschnitten ist, die wie Programme ausführbar sind, ist die zusätzliche Angabe CONNECTION-SCOPE=*PROGRAM anzuraten.

START-ADDRESS =
Legt die Anfangsadresse im Klasse-6-Speicher fest.

START-ADDRESS = *ANY
Voreinstellung: die Lage des Subsystems im Klasse-6-Speicher wird von DSSM festgelegt.

START-ADDRESS = <x-string 7..8>
Start-Adresse im Segment-Raster, an der die Anfangsadresse des Subsystems liegen soll. Als Wert ist eine 8-stellige Hexadezimalkonstante anzugeben, die ein Vielfaches von X'100000' sein muss.

MEMORY-CLASS = *BY-SLICE(...)
Das angegebene Subsystem ist ein nicht-privilegiertes Subsystem und besteht aus einem LLM, das aus einem mehrbenutzbaren Code (Programmbereich) und einem nichtmehrbenutzbaren Code (Datenbereich) besteht.

Der Programmbereich wird in den gemeinsam benutzbaren Adressraum geladen (das entspricht MEMORY-CLASS=*SYSTEM-GLOBAL). Der Datenbereich wird in den Benutzeradressraum der Holdertask geladen und in die privaten Benutzeradressräume der angeschlossenen Tasks an die selbe Adresse kopiert. Gemeinsam mit MEMORY-CLASS=*BY-SLICE müssen folgende Werte angegeben werden:

SUBSYSTEM-LOAD-MODE=*ADVANCED und CONNECTION-ACCESS=*ALL.

Der Wert kann nur zusammen angegeben werden mit den Operanden MODE=*LINK und CONNECTION-SCOPE=*TASK oder *PROGRAM.

SIZE = <integer 1..32767>
Größe des angeforderten Speicherplatzes für den Datenbereich in 4K-Seiten.
Der Wert ist mindestens so groß zu wählen, dass der Datenbereich und evtl. vom Subsystem in den Benutzeradressraum nachzuladende Selectable Units/Load Units in vollem Umfang geladen werden können.
Die obere Grenze ist generierungsabhängig.

LINK-ENTRY = <text 1..8 without-sep>(...)
Definiert den Namen des zum Laden benötigten Bindemoduls/ENTRY/CSECT (als Operand im Makroaufruf $PBBND1 an den dynamischen Bindelader DBL). Das Subsystem muss von diesem ENTRY vollständig (ggf. per Autolink) geladen werden.

AUTOLINK =
Steuert den Aufruf der Autolink-Funktion beim Binden und Laden.
Die Autolink-Funktion des Binders ermöglicht das automatische Einfügen von Modulen, die nicht mit entsprechenden Anweisungen explizit eingefügt werden.
Die Funktion erspart vor allem den Benutzern der höheren Programmiersprachen, die zahlreich benötigten Module des Laufzeitsystems (= Run Time System) mit expliziten Anweisungen einzeln einzufügen. Ein nähere Beschreibung der Autolink-Funktion ist im Handbuch „BLSSERV“ zu finden.

Die Autolink-Funktion kann auch implizit umgangen werden, wenn während des Bindens des zu ladenden Objektmoduls der erste Externverweis auf ein vorgebundenes Großmodul zielt. Der Vorteil dieser Vorgehensweise ergibt sich daraus, dass das Seitenwechselverhalten bei der späteren Ausführung bereits im Vorfeld (während des Bindens) optimiert werden kann. Zudem können Fehler während des Bindens auf diese Weise vermieden werden.

AUTOLINK = *FORBIDDEN /*ALLOWED
Voreinstellung: Die Autolink-Funktion wird unterdrückt oder zugelassen.

REFERENCED-SUBSYSTEM =
Legt fest, ob es eine Liste von Subsystemen gibt, zu denen Adressbeziehungen bestehen und die zur Auflösung von Externverweisen benutzt werden.

REFERENCED-SUBSYSTEM = *NONE
Voreinstellung: das zu definierende Subsystem besitzt keine Externverweise.

REFERENCED-SUBSYSTEM = list-poss(15): <structured-name 1..8>
Das zu definierende Subsystem besitzt Externverweise auf maximal 15 andere Subsysteme, die zur Auflösung dieser Externverweise benutzt werden müssen. Fehlt eines der hier genannten Subsysteme beim Aktivieren oder Deaktivieren (und ist gleichzeitig eine Überprüfung der Externverweise mit dem Operanden CHECK-REFERENCE=*YES angefordert), wird die Aktion abgebrochen.
Auch das „Basis-Subsystem“ des BS2000, das Control Program, kann über diese Externverweise - mit dem Namen CP - angesprochen werden. Es kann in der folgenden Unterstruktur sowohl genau eine Version spezifiziert werden, als auch ein Bereich von Versionen angegeben werden, innerhalb dessen alle Versionen herangezogen werden sollen.

Wird die Version des Subsystems CP über eine Versionsbereichsliste eingegrenzt, prüft DSSM die Kompatibilität zwischen der aktuellen CP-Version und Versionen in der Bereichsliste. Nur wenn die Versionen kompatibel sind, wird das Subsystem geladen.

Auf folgende Einschränkungen bei der Angabe von Subsystemen, zu denen Adressbeziehungen bestehen, ist zu achten:

- Es dürfen keine Adressbeziehungen zu Subsystemen vereinbart werden, die das Attribut MEMORY-CLASS=*LOCAL-PRIVILEGED/*LOCAL-UNPRIVILEGED/*BY-SLICE besitzen.
- Subsysteme, die das Attribut STOP-AT-SHUTDOWN=*NO besitzen, dürfen Adressbeziehungen nur zu solchen Subsystemen aufweisen, die auch dieses Attribut besitzen.
- Ist für das zu definierende Subsystem das Attribut SUBSYSTEM-ACCESS=*SYSTEM vergeben, dürfen keine Subsysteme angesprochen werden, die mit SUBSYSTEM-ACCESS =*LOW bzw. SUBSYSTEM-ACCESS=*HIGH definiert sind.
- Ein nicht-privilegiertes Subsystem darf generell keine Adressbeziehung zum Control Program CP aufweisen.
- Wird auf ein Subsystem verwiesen, das zumindest eine Version besitzt, die im Coexistence- oder Exchange-Modus betrieben werden darf, ist die eindeutige Version anzugeben.
- Die Adressbeziehungen müssen in Abhängigkeit der Startattribute (Operand CREATION-TIME) definiert werden; d.h. Subsysteme dürfen Beziehungen nur zu solchen Subsystemen aufnehmen, deren Start zeitgleich oder früher erfolgt.

UNRESOLVED-EXTERNALS =
Vereinbart das Verhalten des Ladevorgangs, wenn Externverweise nicht aufgelöst werden können.

UNRESOLVED-EXTERNALS = *FORBIDDEN
Voreinstellung: der Ladevorgang wird bei nicht auflösbaren Externverweisen abgebrochen.

UNRESOLVED-EXTERNALS = *ALLOWED
Der Ladevorgang wird fortgesetzt; nicht auflösbare Externverweise werden mit dem Wert X'FFFFFFFF' besetzt.

CHECK-REFERENCE =
Vereinbart, ob die im Operanden REFERENCED-SUBSYSTEM angegebenen Subsysteme im Hinblick auf deren Zustand und Verfügbarkeit hin überprüft werden sollen.

CHECK-REFERENCE = *YES
Voreinstellung: die Referenz-Subsysteme werden überprüft. Fehlt eines der Subsysteme, bricht DSSM das Aktivieren oder Entladen des Subsystems ab.

CHECK-REFERENCE = *NO
DSSM soll keine Prüfung vornehmen.
Generiert der Anwender allerdings mit dieser Anweisung auch komplexe Subsysteme, führt DSSM die geforderten Funktionen **trotz** möglicher Konflikte durch:

- Das Kommando START-SUBSYSTEM lädt das angegebene Subsystem, auch wenn ein Subsystem, zu dem definierte Beziehungen bestehen, noch nicht vollständig geladen ist.
- Die Kommandos RESUME-SUBSYSTEM, STOP-SUBSYSTEM oder HOLD-SUBSYSTEM werden ohne Prüfung von Beziehungen und Abhängigkeiten von DSSM ausgeführt.

RELATED-SUBSYSTEM =
Legt fest, ob es eine Liste von Subsystemen gibt, zu denen Abhängigkeitsbeziehungen bestehen.

RELATED-SUBSYSTEM = *NONE
Voreinstellung: das zu definierende Subsysteme besitzt keine Abhängigkeitsbeziehungen.

RELATED-SUBSYSTEM = list-poss(100): <structured-name 1..8>
Das zu definierende Subsystem besitzt Abhängigkeitsbeziehungen zu maximal 100 anderen Subsystemen, ohne die das zu definierende Subsystem nicht funktionsfähig ist.Abhängigkeitsbeziehungen dürfen auch zum BS2000 Control Program CP definiert
werden. Die Regeln und Einschränkungen, die hierbei zu beachten sind, gelten analog für Adressbeziehungen (Operand REFERENCED-SUBSYSTEM) und sind dort näher erläutert.Die Abhängigkeitsbeziehungen zielen jeweils auf eine Version eines Subsystems. In der folgenden Unterstruktur kann sowohl genau eine Version spezifiziert werden, als auch ein Bereich von Versionen angegeben werden, innerhalb dessen alle Versionen herangezogen werden sollen.

Auf folgende Einschränkungen ist allgemein bei der Angabe von abhängigen Subsystemen ist zu achten:

- Die definierte Beziehung muss zyklenfrei sein. Ein Zyklus entsteht, wenn Subsystem A abhängig von B, B abhängig von C und dieses wieder abhängig von A ist.
- Ist für das zu definierende Subsystem das Attribut MEMORY-CLASS=*SYSTEM-GLOBAL vergeben, dürfen keine Subsysteme angesprochen werden, die mit MEMORY-CLASS=*LOCAL-PRIVILEGED oder *LOCAL-UNPRIVILEGED definiert sind.
- Für Subsysteme, die das Attribut SUBSYSTEM-ACCESS=*SYSTEM besitzen, darf keine Abhängigkeitsbeziehung zu Subsystemen definiert werden, für die SUBSYSTEM-ACCESS=*LOW bzw. SUBSYSTEM-ACCESS=*HIGH oder MEMORY-CLASS=*BY-SLICE gilt.
- Die Abhängigkeitsbeziehungen müssen entsprechend den Startattributen (Operand CREATION-TIME) definiert werden; d.h. ein Subsystem darf nur von solchen Subsystemen abhängen, deren Start zeitgleich oder früher erfolgt.

  LOWEST-VERSION =
  Spezifiziert den unteren Wert (niedrigste Version) innerhalb der Versionsbereichsliste der Subsysteme.

  LOWEST-VERSION = *LOWEST-EXISTING
  Die niedrigste Version im Katalog soll angesprochen werden.

  LOWEST-VERSION = <c-string 3..8> / <text 3..8>
  Version eines Subsystems, die als Untergrenze des Versionsbereiches fungieren soll.

  HIGHEST-VERSION =
  Spezifiziert den oberen Wert (höchste Version) innerhalb der Versionsbereichsliste der Subsysteme.

HIGHEST-VERSION = *HIGHEST-EXISTING

Die höchste Version im Katalog soll angesprochen werden.

HIGHEST-VERSION = <c-string 3..8> / <text 3..8>

Version eines Subsystems, die als Obergrenze des Versionsbereiches fungieren soll.

4.3.14 SHOW-CATALOG

Subsystem-Konfiguration anzeigen

Funktionsbeschreibung

Mit dieser Anweisung wird der Inhalt einer Subsystem-Konfiguration, die in einem Subsystemkatalog gespeichert ist, wahlweise auf Bildschirm oder in eine Datei ausgegeben.
Die Ausgabe beinhaltet immer den Namen des Subsystemkatalogs und die Information, ob und wie im Fehlerfall der Katalog abgespeichert wird (dieses Verhalten wird in der Anweisung SAVE-CATALOG ..., FORCED=... eingestellt).

SHOW-CATALOG wird zurückgewiesen, wenn der angegebene Datei- oder Subsystemname oder die angegebene Subsystemversion nicht existiert. Ist der Subsystemkatalog ohne Inhalt, wird eine Warnung ausgegeben und die Anweisung abgebrochen.

Wenn der angegebene Dateiname nicht dem des aktuell eröffneten Kataloges entspricht, wird folgende Meldung ausgegeben:

```
SCM0011    WOLLEN SIE WIRKLICH DEN KATALOG IM SPEICHER ÜBERSCHREIBEN ? (Y/N)
```

Bei Antwort **Y** gehen die virtuellen Definitionen des aktuellen Kataloges verloren.
Bei Antwort **N** wird die Ausführung der Anweisung SHOW-CATALOG abgebrochen. Der Anwender kann mit SAVE-CATALOG die bisher nicht gespeicherten Subsystemdefinitionen in einer Datei ablegen.

Wurde im Vorfeld die Anweisung CHECK-CATALOG nicht zur Konsistenzprüfung eingesetzt, kann SHOW-CATALOG Fehler bzgl. Konsistenz im Subsystemkatalog aufdecken.

Format

SHOW-CATALOG

CATALOG-NAME = ***CURR**ENT / <filename 1..54 without-gen-vers>

,**CONTAINED-SUBSYSTEMS** = ***NO** / ***Y**ES

,**SUBSYS**TEM-NAME = ***ALL** / <structured-name 1..8>(...)

<structured-name 1..8>(...)

VERSION = ***ALL** / <c-string 3..8> / <text 3..8>

,**GENERAL-ATTRIBUTES** = ***Y**ES / ***NO**

,**INTERNAL-ENTRIES** = ***Y**ES / ***NO**

,**MEMORY-ATTRIBUTES** = ***Y**ES / ***NO**

,**RELATED-FILES** = ***Y**ES / ***NO**

,**LINK-ATTRIBUTES** = ***Y**ES / ***NO**

,**REFERENCE-RELATION** = ***Y**ES / ***NO**

,**DEPENDENCE-RELATION** = ***Y**ES / ***NO**

,**ADDR-SPACE-RELATION** = ***Y**ES / ***NO**

,**HOLDER-TASK-INFO** = ***Y**ES / ***NO**

,**SUBSYSTEM-ENTRIES** = ***Y**ES / ***NO**

,**OUTPUT** = ***SYSLST** (...) / ***SYSOUT**

***SYSLST**(...)

SYSLST-NUMBER = ***STD** / <integer 1..99>

,**SUMM**ARY = ***Y**ES / ***NO**

Operandenbeschreibung

CATALOG-NAME =
Vereinbart den Namen des Kataloges, in dem die anzuzeigenden Definitionen hinterlegt sind. Existiert der Katalog-Dateiname nicht, oder ist der angegebene Katalog ohne Inhalt, wird die Anweisung zurückgewiesen.

CATALOG-NAME = *CURRENT
Voreinstellung: der Inhalt des aktuell geöffneten Kataloges (Anweisung START-CATALOG-CREATION bzw. START-CATALOG-MODIFICATION) soll ausgegeben werden.

CATALOG-NAME = <filename 1..54 without-gen-vers>
Vollqualifizierter Name des statischen Subsystemkataloges, dessen Inhalt angezeigt werden soll.

CONTAINED-SUBSYSTEMS = *NO / *YES
Vereinbart, ob die Liste der DSSM-Subsysteme, die als Definitionen im Katalog enthalten sind, ausgegeben werden soll (*YES) oder nicht (*NO).

SUBSYSTEM-NAME =
Vereinbart, über welche Subsysteme Informationen angefordert werden.
Wenn der Name des Subsystems oder eine Version, über die Informationen angefordert werden, nicht existiert, wird die Anweisung zurückgewiesen. Der Umfang der Informationen, die über die Subsysteme ausgegeben werden sollen, kann in den folgenden Operanden durch Angabe entsprechender Kriterien eingeschränkt werden.

SUBSYSTEM-NAME = *ALL
Voreinstellung: es sollen Informationen über alle Subsysteme angefordert werden, die im Katalog verzeichnet sind.

SUBSYSTEM-NAME = <structured-name 1..8>(...)
Name des Subsystems, über das SSCM die Informationen aus dem Katalog bereitstellt.

VERSION =
Spezifiziert die Version des ausgewählten Subsystems.

VERSION = *ALL
Alle im Katalog gespeicherten Versionen des Subsystems sollen in die Informationsausgabe eingeschlossen werden.

VERSION = <c-string 3..8> / <text 3..8>
Nur über diese Version des ausgewählten Subsystems sollen Informationen aus dem Katalog bereitgestellt werden.

GENERAL-ATTRIBUTES = *YES / *NO
Vereinbart, ob folgende generelle Eigenschaften der genannten Subsysteme aus dem Katalog gelesen werden sollen (*YES) oder nicht (*NO):

- Wann soll das Subsystem nach Systemeinleitung gestartet werden? (CREATION-TIME)
- In welchem Lademodus soll das Subsystem geladen werden? (SUBSYSTEM-LOAD-MODE)
- Soll das Subsystem bei Shutdown automisch entladen werden? (STOP-AT-SHUTDOWN)
- Darf das geladene Subsystem angehalten oder entladen werden? (SUBSYSTEM-HOLD)
- Dürfen die Kommandos zur Steuerung des Subsystems verwendet werden? (STATE-CHANGE-CMDS)
- Ist die FORCE-Option zugelassen? (FORCED-STATE-CHANGE)
- Ist die RESET-Option zugelassen? (RESET)
- Muss bei abnormaler Beendigung der Holdertask die Initialisierungsroutine durchlaufen werden? (RESTART-REQUIRED)
- Darf mehr als eine Version des Subsystems gleichzeitig aktiv sein? (VERSION-COEXISTENCE)
- Dürfen zwei Versionen eines Subsystems dynamisch ausgetauscht werden? (VERSION-EXCHANGE)
- Wie lautet der Name der INSTALLATION-UNIT des Subsystems? (INSTALLATION-UNIT)

- Wie lautet das Copyright (Text und Datum) des Subsystems?
  (COPYRIGHT)

INTERNAL-ENTRIES = *YES / *NO
Vereinbart, ob folgende Informationen über die Einsprungstellen der angegebenen Subsysteme von SSCM bereitgestellt werden sollen (*YES) oder nicht (*NO):

- die Namen der Einsprungstellen für die Subsystemroutinen INIT-, STOPCOM-, DEINIT- und CLOSE-CTRL-Routine
- der Name der Einsprungstelle, die für die dynamische Identitätsprüfung herangezogen wird (DYNAMIC-CHECK-ENTRY)
- der Name der Schnittstellenversion für den Aufruf der INIT-, STOPCOM-, DEINIT- oder CLOSE-CTRL-Routinen (INTERFACE-VERSION)

MEMORY-ATTRIBUTES = *YES / *NO
Vereinbart, ob folgende Speicher-relevanten Informationen, die im Katalog über die Subsysteme gespeichert sind, ausgegeben werden sollen (*YES) oder nicht (*NO):

- Speicherklasse (MEMORY-CLASS)
- Größe des benötigten Adressraums (SIZE)
- Anfangsadresse des Subsystemcodes (START-ADDRESS)
- Privilegierung und Zugriffsberechtigung bezüglich Adressraum (SUBSYSTEM-ACCESS)

RELATED-FILES = *YES / *NO
Vereinbart, ob Informationen über die Nebenkomponenten des Subsystems geliefert werden sollen (*YES) oder nicht (*NO). In die Ausgabe eingeschlossen ist die Information, ob die Verwendung einer Rep-Datei für dieses Subsystem verpflichtend ist (REP-FILE-MANDATORY) und unter welcher Benutzerkennung die Nebenkomponenten katalogisiert sind (INSTALLATION-USERID).

Unter dem Begriff Nebenkomponenten werden zusammengefasst:

- die Objektmoduldatei des Subsystems (LIBRARY)
- die Meldungsdatei (MESSAGE-FILE)
- die Syntaxdatei (SYNTAX-FILE)
- die Informationsdatei des Subsystems (SUBSYSTEM-INFO-FILE)
- die Rep-Datei (REP-FILE)

LINK-ATTRIBUTES = *YES / *NO
Vereinbart, ob die Informationen aus dem Katalog zu lesen sind (*YES) oder nicht zu lesen sind (*NO), die in Bezug auf das Binden und Laden des Subsystems gespeichert wurden:

- der Name des zum Laden benötigten Bindemoduls/ENTRY/CSECT (LINK-ENTRY)
- die Einbindung der Autolink-Funktion (AUTOLINK)
- die Informationen über das Verhalten bei nicht auflösbaren Externverweisen (UNRESOLVED)
- die Einbindung des Prüflaufs für Referenz-Subsysteme (CHECK-REFERENCE)

REFERENCE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, zu denen Adressbeziehungen bestehen, bei der Ausgabe der Katalog-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

DEPENDENCE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, zu denen Abhängigkeitsbeziehungen bestehen, bei der Ausgabe der Katalog-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

ADDR-SPACE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, mit denen eine Adressraumüberschneidung vermieden werden muss, bei der Ausgabe der Katalog-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

HOLDER-TASK-INFO = *YES / *NO
Vereinbart, ob die Identifikation der Holdertask und die Liste der Subsysteme, die in einer gemeinsamen Holdertask anzulegen sind, bei der Ausgabe der Katalog-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

SUBSYSTEM-ENTRIES = *YES / *NO
Vereinbart, ob die Liste der bei der Definition des Subsystems vereinbarten Auftragseingänge und deren folgende Eigenschaften aus dem Katalog gelesen werden soll (*YES) oder nicht (*NO):

- Typ des vereinbarten Auftragseingangs (MODE)
- Nummer der Routine (bei SVC oder System-Exit) (NUMBER)
- die Funktionsnummer der Einsprungstelle (FUNCTION-NUMBER)
- die Version der Funktionsnummer (FUNCTION-VERSION)
- die Information über Aufruf durch System-Exit-Routinen (CALL-BY-SYSTEM-EXIT)
- die Privilegierung und Zugriffsberechtigung bezüglich Einsprungstellen (CONNECTION-ACCESS und CONNECTION-SCOPE)

OUTPUT =
Vereinbart, wohin die von der Anweisung generierten Informationen ausgegeben werden sollen.

OUTPUT = *SYSLST(...)
Voreinstellung: die Meldungen sollen nach SYSLST ausgegeben werden.

> **SYSLST-NUMBER =**
> Bezeichnet die SYSLST-Datei, in die die Ausgabe erfolgen soll.
>
> **SYSLST-NUMBER = *STD**
> Voreinstellung: die Ausgabe soll in die Standard-Systemdatei SYSLST erfolgen.
>
> **SYSLST-NUMBER = <integer 1..99>**
> Die Ausgabe soll in eine Systemdatei aus der Menge SYSLST01 bis SYSLST99 erfolgen, deren Nummer hier anzugeben ist.
>
> **SUMMARY = *YES / *NO**
> Legt fest, ob eine zusammenfassende Übersicht über alle Subsysteme, die im Katalog verzeichnet sind, der geforderten Ausgabe vorangestellt werden soll (*YES) oder nicht (*NO). Für OUTPUT=*SYSLST (SUMMARY=*YES) siehe unten: Abkürzungs-Übersicht.

OUTPUT = *SYSOUT
Die Meldungen werden auf der Datensichtstation ausgegeben.

Bei der Ausgabe einer **Übersicht** über alle Subsysteme im Katalog (siehe Operand

OUTPUT=*SYSLST(SUMMARY=*YES)) werden folgende Abkürzungen verwendet:

- für CREATION-TIME
  - ACR: *AT-CREATION-REQUEST
  - ASC: *AT-SUBSYSTEM-CALL
  - ADL: *AT-DSSM-LOAD
  - BDL: *BEFORE-DSSM-LOAD
  - MAS: *MANDATORY-AT-STARTUP
  - BSR: *BEFORE-SYSTEM-READY
  - ASR: *AFTER-SYSTEM-READY
- für MEMORY-CLASS
  - S: *SYSTEM-GLOBAL
  - P: *LOCAL-PRIVILEGED
  - U: *LOCAL-UNPRIVILEGED
  - B: *BY-SLICE
- für SUBSYSTEM-ACCESS
  - SYS: *SYSTEM
  - ALL:: *LOW / *HIGH
- für INTERNAL-ENTRIES
  - DYN: *DYNAMIC
  - YES: name
  - NO: *NO
- für CONNECTION-ACCESS
  - SYS: *SYSTEM
- für STATE-CHANGE-CMDS
  - ADM: *BY-ADMINISTRATOR-ONLY
- für REP-FILE
  - MAN: REP-FILE = *STD / dateiname und REP-FILE-MANDATORY = *YES
- für GENERAL-ATTRIBUTES
  - YES: *ALLOWED
  - NO: *FORBIDDEN
- für RELATED-FILES
  - YES: *STD / dateiname
  - NO: *NO
  - IMO: *INSTALLED

4.3.15 SHOW-SSD

Inhalt eines SSD-Objects (Subsystemdefinitionen) anzeigen

Funktionsbeschreibung

Mit dieser Anweisung wird der Inhalt eines SSD-Objects wahlweise auf Bildschirm oder in eine andere Datei ausgegeben. In einem SSD-Object sind die Definitionen eines oder mehrerer Subsysteme hinterlegt.
Die Ausgabe erfolgt separat für alle Subsystemdefinitionen, die im angegebenen SSD-Object (ISAM-Datei) gespeichert sind.

Name und Version des SSD-Objects werden immer ausgegeben, ebenso der Name der Domäne für das SSD-Object und ob PULS-Problemmeldungen in das SSD-Object eingearbeitet sind.

Es ist zu beachten, dass SHOW-SSD nicht den vollständigen Inhalt des SSD-Objects anzeigt, sondern nur die Anweisungen zur Subsystemdefinition, die nach der letzten Anweisung ADD-CATALOG-ENTRY eingegeben wurden.

Format

SHOW-SSD

SSD-FILE-NAME = ***CURR** ENT / <filename 1..54 without-gen-vers>

,**GENERAL-ATTRIBUTES** = ***Y** ES / ***NO**

,**INTERNAL-ENTRIES** = ***Y** ES / ***NO**

,**MEMORY-ATTRIBUTES** = ***Y** ES / ***NO**

,**RELATED-FILES** = ***Y** ES / ***NO**

,**LINK-ATTRIBUTES** = ***Y** ES / ***NO**

,**REFERENCE-RELATION** = ***Y** ES / ***NO**

,**DEPENDENCE-RELATION** = ***Y** ES / ***NO**

,**ADDR-SPACE-RELATION** = ***Y** ES / ***NO**

,**HOLDER-TASK-INFO** = ***Y** ES / ***NO**

,**SUBSYSTEM-ENTRIES** = ***Y** ES / ***NO**

,**OUTPUT** = ***SYSLST** (...) / ***SYSOUT**

***SYSLST**(...)

SYSLST-NUMBER = ***STD** / <integer 1..99>

Operandenbeschreibung

SSD-FILE-NAME =
Gibt den Namen des SSD-Objects (ISAM-Datei) an, in dem die anzuzeigenden Definitionen hinterlegt sind. Existiert keine ISAM-Datei mit diesem Namen oder ist die angegebene Datei ohne Inhalt, wird die Anweisung zurückgewiesen.

SSD-FILE-NAME = *CURRENT
Voreinstellung: der Inhalt des aktuell geöffneten SSD-Objects (Anweisung START-SSD-CREATION) soll ausgegeben werden.

SSD-FILE-NAME = <filename 1..54 without-gen-vers>
Vollqualifizierter Name der SSD-Objects, dessen Inhalt angezeigt werden soll.

GENERAL-ATTRIBUTES = *YES / *NO
Vereinbart, ob folgende generelle Eigenschaften der SSD-Object definierten Subsysteme angezeigt werden sollen (*YES) oder nicht (*NO):

- Wann soll das Subsystem nach Systemeinleitung gestartet werden?
  (CREATION-TIME)
- In welchem Lademodus soll das Subsystem geladen werden?
  (SUBSYSTEM-LOAD-MODE)
- Soll das Subsystem bei Shutdown automisch entladen werden?
  (STOP-AT-SHUTDOWN)
- Darf das geladene Subsystem angehalten oder entladen werden?
  (SUBSYSTEM-HOLD)
- Dürfen die Kommandos zur Steuerung des Subsystems verwendet werden?
  (STATE-CHANGE-CMDS)
- Ist die FORCE-Option zugelassen?
  (FORCED-STATE-CHANGE)
- Ist die RESET-Option zugelassen?
  (RESET)
- Muss bei abnormaler Beendigung der Holdertask die Initialisierungsroutine durchlaufen werden?
  (RESTART-REQUIRED)
- Darf mehr als eine Version des Subsystems gleichzeitig aktiv sein?
  (VERSION-COEXISTENCE)
- Dürfen zwei Versionen eines Subsystems dynamisch ausgetauscht werden?
  (VERSION-EXCHANGE)
- Wie lautet der Name der INSTALLATION-UNIT des Subsystems?
  (INSTALLATION-UNIT)
- Wie lautet das Copyright (Text und Datum) des Subsystems?
  (COPYRIGHT)

INTERNAL-ENTRIES = *YES / *NO
Vereinbart, ob folgende Informationen über die Einsprungstellen der enthaltenen Subsysteme von SSCM bereitgestellt werden sollen (*YES) oder nicht (*NO):

- die Namen der Einsprungstellen für die Subsystemroutinen INIT-, STOPCOM-, DEINIT- und CLOSE-CTRL-Routine.
- der Name der Einsprungstelle, die für die dynamische Identitätsprüfung herangezogen wird (DYNAMIC-CHECK-ENTRY)
- der Name der Schnittstellenversion für den Aufruf der INIT-, STOPCOM-, DEINIT- oder CLOSE-CTRL-Routinen (INTERFACE-VERSION)

MEMORY-ATTRIBUTES = *YES / *NO
Vereinbart, ob folgende Speicher-relevanten Informationen, die als Teil der Subsystemdefinition SSD-Object gespeichert sind, ausgegeben werden sollen (*YES) oder nicht (*NO):

- Speicherklasse (MEMORY-CLASS)
- Größe des benötigten Adressraums (SIZE)
- Anfangsadresse des Subsystemcodes (START-ADDRESS)
- Privilegierung und Zugriffsberechtigung bzgl. Adressraum (SUBSYSTEM-ACCESS)

RELATED-FILES = *YES / *NO
Vereinbart, ob Informationen über die Nebenkomponenten des Subsystems geliefert werden sollen (*YES) oder nicht (*NO). In die Ausgabe eingeschlossen ist die Information, ob die Verwendung einer Rep-Datei für dieses Subsystem verpflichtend ist (REP-FILE-MANDATORY) und unter welcher Benutzerkennung die Nebenkomponenten katalogisiert sind (INSTALLATION-USERID). Unter dem Begriff Nebenkomponenten werden zusammengefasst:

- die Objektmoduldatei des Subsystems (LIBRARY)
- die Meldungsdatei (MESSAGE-FILE)
- die Syntaxdatei (SYNTAX-FILE)
- die Informationsdatei des Subsystems (SUBSYSTEM-INFO-FILE)
- die Rep-Datei (REP-FILE)

LINK-ATTRIBUTES = *YES / *NO
Vereinbart, ob die Informationen aus dem SSD-Object zu lesen sind (*YES) oder nicht (*NO), die in Bezug auf das Binden und Laden des Subsystems gespeichert sind:

- der Name des zum Laden benötigten Bindemoduls/ENTRY/CSECT (LINK-ENTRY)
- die Einbindung der Autolink-Funktion (AUTOLINK)
- die Informationen über das Verhalten bei nicht auflösbaren Externverweisen (UNRESOLVED)
- die Einbindung des Prüflaufs für Referenz-Subsysteme (CHECK-REFERENCE)

REFRENCE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, zu denen Adressbeziehungen bestehen, bei der Ausgabe der SSD-Datei-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

DEPENDENCE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, zu denen Abhängigkeitsbeziehungen bestehen, bei der Ausgabe der SSD-Object-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

ADDR-SPACE-RELATION = *YES / *NO
Vereinbart, ob die Liste der Subsysteme, mit denen eine Adressraumüberschneidung vermieden werden muss, bei der Ausgabe der SSD-Object-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

HOLDER-TASK-INFO = *YES / *NO
Vereinbart, ob die Identifikation der Holdertask und die Liste der Subsysteme, die in einer gemeinsamen Holdertask anzulegen sind, bei der Ausgabe der SSD-Object-Informationen berücksichtigt werden soll (*YES) oder nicht (*NO).

SUBSYSTEM-ENTRIES = *YES / *NO
Vereinbart, ob die Liste der bei der Definition des Subsystems vereinbarten Auftragseingänge und deren folgende Eigenschaften aus dem SSD-Object gelesen werden soll (*YES) oder nicht (*NO):

- Typ des vereinbarten Auftragseingangs (MODE)
- Nummer der Routine (bei SVC oder System-Exit) (NUMBER)
- die Funktionsnummer der Einsprungstelle (FUNCTION-NUMBER)
- die Version der Funktionsnummer (FUNCTION-VERSION)
- die Information über Aufruf durch System-Exit-Routinen (CALL-BY-SYSTEM-EXIT)
- die Privilegierung und Zugriffsberechtigung bezüglich Einsprungstellen (CONNECTION-ACCESS und CONNECTION-SCOPE)

OUTPUT =
Vereinbart den Ausgabeort der von der Anweisung generierten Informationen.

OUTPUT = *SYSLST(...)
Voreinstellung: die Meldungen sollen nach SYSLST ausgegeben werden.

SYSLST-NUMBER =
Bezeichnet die SYSLST-Datei, in die die Ausgabe erfolgen soll.

SYSLST-NUMBER = *STD
Voreinstellung: die Ausgabe soll in die Standard-Systemdatei SYSLST erfolgen.

SYSLST-NUMBER = <integer 1..99>
Die Ausgabe soll in eine Systemdatei aus der Menge SYSLST01 bis SYSLST99 erfolgen, deren Nummer hier anzugeben ist.

OUTPUT = SYSOUT
Die Meldungen werden auf der Datensichtstation ausgegeben.

4.3.16 START-CATALOG-CREATION

Name eines statischen Subsystemkatalogs vereinbaren

Funktionsbeschreibung

Mit dieser Anweisung wird der Name eines neuen statischen Subsystemkataloges vereinbart. In diese Datei können dann die SSD-Objects, die die Definitionen der Subsysteme beinhalten, eingebunden werden.

START-CATALOG-CREATION wird mit einer Fehlermeldung abgewiesen, wenn eine Datei gleichen Namens bereits existiert oder zuvor ein gleich lautender Katalogeintrag mit CREATE-FILE erzeugt wurde. Die Definition wird mit der Sicherung des neuen Kataloges (Anweisung SAVE-CATALOG) abgeschlossen.

Folgen allerdings zwei Anweisungen START-CATALOG-CREATION oder START-CATALOG-MODIFICATION aufeinander, erscheint in Dialogaufträgen die Meldung:

```
SCM0011   WOLLEN SIE WIRKLICH DEN KATALOG IM SPEICHER ÜBERSCHREIBEN? (Y/N)
```

Auf die Antwort **Y** wird der belegte Speicherplatz freigegeben und zuvor vereinbarte Eigenschaften des Katalogs werden verworfen.
Auf die Antwort **N** wird die zweite START-CATALOG-CREATION-Anweisung ungültig und die erste Anweisung kann mit SAVE-CATALOG abgeschlossen werden.

Bei Stapelaufträgen wird implizit die Antwort **Y** gegeben.

Format

START-CATALOG-CREATION
CATALOG-NAME = ***STD** / <filename 1..51 without-gen-vers>

Operandenbeschreibung

CATALOG-NAME =
Name des zu erzeugenden statischen Subsystemkataloges.

Existiert kein Katalog mit diesem Namen, wird die Anweisung abgewiesen.

CATALOG-NAME = *STD
Voreinstellung ist die Datei SYS.SSD.CAT.X auf dem Home-Pubset.

CATALOG-NAME = <filename 1..51 without-gen-vers>
Vollqualifizierter Dateiname.

4.3.17 START-CATALOG-MODIFICATION

Statischen Subsystemkatalog verändern

Funktionsbeschreibung

Mit dieser Anweisung kann ein bereits vorhandener statischer Subsystemkatalog modifiziert werden. In diese Datei können dann neue SSD-Objects, die die Definitionen der Subsysteme beinhalten, eingebunden werden.

Die Definition eines veränderten Katalogs wird mit den Anweisungen CHECK-CATALOG und SAVE-CATALOG abgeschlossen.

START-CATALOG-MODIFICATION wird abgewiesen, wenn der Name der angegebenen Datei nicht existiert oder zuvor nicht mit der Anweisung START-CATALOG-CREATION erzeugt wurde.

Folgen zwei Anweisungen START-CATALOG-MODIFICATION aufeinander oder wird START-CATALOG-MODIFICATION nach der Anweisung START-CATALOG-CREATION angegeben ohne ein abschließendes SAVE-CATALOG, erscheint in Dialogaufträgen die Meldung:

```
SCM0011  WOLLEN SIE WIRKLICH DEN KATALOG IM SPEICHER ÜBERSCHREIBEN ? (Y/N)
```

Auf die Antwort **Y** wird der belegte Speicherplatz freigegeben und zuvor veränderte Eigenschaften des Kataloges werden verworfen.
Auf die Antwort **N** wird die zweite START-CATALOG-MODIFICATION-Anweisung ungültig und die erste Anweisung kann mit SAVE-CATALOG abgeschlossen werden.

Bei Stapelaufträgen wird implizit die Antwort **Y** gegeben.

Format

START-CATALOG-MODIFICATION
CATALOG-NAME = <filename 1..54 without-gen-vers>

Operandenbeschreibung

CATALOG-NAME = <filename 1..54 without-gen-vers>
Vollqualifizierter Dateiname des Subsystemkataloges, der verändert werden soll.

4.3.18 START-SSD-CREATION

SSD-Object zur Aufnahme von Subsystemdefinitionen generieren

Funktionsbeschreibung

Mit der Anweisung START-SSD-CREATION wird die Generierung eines SSD-Objects zur Aufnahme von Subsystemdefinitionen angestoßen. Der Anwender muss den Namen des SSD-Objects, den Namen der Datei, in der es gespeichert werden soll sowie die Namen der vom Subsystem benötigten/referenzierten Dateien angeben.

START-SSD-CREATION wird mit einer Fehlermeldung abgewiesen, wenn die bei SSD-FILE-NAME anzugebende Datei bereits existiert.

Die Anweisung SAVE-SSD zur Sicherung aller vereinbarten Eigenschaften kann nur nach erfolgreicher Abarbeitung der Anweisung SET-SUBSYSTEM-ATTRIBUTES eingegeben werden.

Format

START-SSD-CREATION
SSD-NAME = <name 1..8>
,**VERSION** = <integer 1..999>
,**DOMAIN** = <structured-name 1..13>
,**CORRECTION** = ***NONE** / list-poss(20): <alphanum-name 8..8>
,**SSD-FILE-NAME** = <filename 1..51 without-userid without-gen-vers>
,**BLOCK-CONTROL-INFO** = ***STD** / ***WITHIN-DATA-BLOCK**

Operandenbeschreibung

SSD-NAME = <name 1..8>
Name des SSD-Objects.

Der Name muss folgender Konvention genügen: die ersten drei Buchstaben entsprechen der Meldungsklasse des Produkts, die letzten drei Buchstaben lauten „SSC“.

VERSION = <integer 1..999>
Version des SSD-Objects.

DOMAIN = <structured-name 1..13>
Bezeichnung der MONSYS-Domäne des SSD-Objects. Über den Namen dieser Domäne ist systemintern eine eindeutige, konsistente Zuordnung aller Komponenten zu dem Subsystem zu realisieren.

CORRECTION =
Gibt an, ob im SSD-Object für die Subsysteme PULS-Problemmeldungen eingearbeitet worden sind.

CORRECTION = *NONE
Voreinstellung: in das SSD-Object wurden keine PULS-Problemmeldungen eingearbeitet.

CORRECTION = list-poss(20) <alphanum-name 8..8>
Liste der maximal 20 Problemmeldungen, die im SSD-Object korrigiert sind.

SSD-FILE-NAME = <filename 1..51 without-userid without-gen-vers>
Name der neu anzulegenden ISAM-Datei, in der das SSD-Object gesichert werden soll.Existiert der Dateiname bereits, wird die Anweisung zurückgewiesen.

BLOCK-CONTROL-INFO =
Gibt an, in welchem Dateiformat die Datei für das SSD-Object angelegt werden soll.

BLOCK-CONTROL-INFO = *STD
Voreinstellung: SSCM versucht zuerst, das SSD-Object als K-Datei (Blockformat PAMKEY) mit der Blocklänge 1 anzulegen (BUFFER-LENGTH=*STD(SIZE=1)).
Im Fehlerfall wird versucht, das SSD-Object mit der Blocklänge 2 anzulegen.
Ist auch dieser Versuch nicht erfolgreich, wird das SSD-Object als NK-Datei mit dem Blockformat DATA und der Blocklänge 2 angelegt.

BLOCK-CONTROL-INFO = *WITHIN-DATA-BLOCK
SSCM legt das SSD-Object als NK-Datei im Blockformat DATA und der Blocklänge 2 an.

Hinweise

Im Anschluss an eine fehlerfreie Bearbeitung der Anweisung START-SSD-CREATION sind nur die folgenden Anweisungen möglich. Die Anweisungen mit (*) können auch mehrmals verwendet werden, um jeweils unterschiedliche Subsysteme (aber nicht unterschiedliche Versionen eines Subsystems) zu definieren.

- ADD-SUBSYSTEM-ENTRIES (*)
- ASSIGN-HOLDER-TASK (*)
- SEPARATE-ADDRESS-SPACE (*)
- SET-SUBSYSTEM-ATTRIBUTES (*)
- SHOW-SSD

4.4 Installation von SSCM

Es wird empfohlen, SSCM mit IMON zu installieren.

Mit SSCM V21.0 werden folgende Dateien ausgeliefert:

- die Modulbibliothek SYSLNK.SSCM.210, die das Bindelademodul SSCM enthält
- die System-Syntaxdatei SYSSDF.SSCM.210 mit Anweisungen für SSCM
- die Benutzer-Syntaxdatei SYSSDF.SSCM.210.USER mit Anweisungen für SSCM
- die Prozedur SYSPRC.SSCM.210, die SSCM mit START-EXECUTABLE-PROGRAM startet
- die Subsystem-Deklarationsdatei SYSSSC.SSCM.210
- die Meldungsdatei SYSMES.SSCM.210
- die Rep-Datei SYSREP.SSCM.210

Folgende Bedingungen müssen für die Installation von SSCM V21.0 erfüllt sein:

1. Die SSCM-Subsystemdefinition muss im Subsystemkatalog aufgenommen sein.
2. Die Bibliothek SYSLNK.SSCM.210 muss unter der Installations-Benutzerkennung katalogisiert sein, die mit ADD-CATALOG-ENTRY oder SET-SUBSYSTEM-ATTRIBUTES bei der Aufnahme von SSCM in den Subsystemkatalog benannt wurde (standardmäßig die System-Benutzerkennung, deren Dateien i.a. mit „$." beginnen).
3. Mit dem Kommando START-SUBSYSTEM SSCM wird das Subsystem SSCM aktiviert und mit dem Kommando START-SSCM gestartet.
   Es gehören auch die Meldungsdatei, die Syntaxdatei und die Rep-Datei dazu; sie werden automatisch aktiviert.
   Die System-Syntaxdatei SYSSDF.SSCM.023 muss das Attribut SHARE=SPECIAL besitzen.

Wird die Prozedur SYSPRC.SSCM.210 zum Aufruf von SSCM V21.0 benutzt, entfallen die Punkte 1 und 3.

Koexistenz von SSCM-Versionen

Die SSCM-Versionen V1.0 bis V21.0 können gleichzeitig im Subsystemkatalog definiert sein. Die Version kann mit dem Kommando SELECT-PRODUCT-VERSION ausgewählt werden. Das Kommando START-SSCM startet dann die gewählte Version.

4.5 Beispiele

Die nachfolgenden Beispiele stellen Folgen von SSCM-Anweisungen dar, die die Vorgehensweise zur Erstellung und Änderung bestimmter Objekte veranschaulichen sollen.

4.5.1 Erstellung eines SSD-Objects

```
/START-SSCM
//START-SSD-CREATION SSD-NAME=SS1SSC,VERSION=001,DOMAIN=DSSM,
        CORRECTION=*NONE,SSD-FILE-NAME=SYSSSC.SS1.001 ————————————————  (1)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS1(VERSION=1.0),
        SUBSYSTEM-ENTRIES=ENTRY1(CONNECTION-ACCESS=*ALL,
                                 CONNECTION-SCOPE=*TASK),
        REFERENCED-SUBSYSTEM=SS2(LOWEST-VERS=01.0,HIGHEST-VERSION=02.5),
        RELATED-SUBSYSTEM=SS3(LOWEST-VERSION=02.0,HIGHEST-VERSION=02.0),
        LINK-ENTRY=E1 ————————————————————————————————————————————————  (2)
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=(SS5,SS6,SS9) ————————————————————————————————  (3)
//ASSIGN-HOLDER-TASK TYPE=*SHARED-HOLDER(
        BY-SUBSYSTEMS=(SS1,SS2,SS3),TSN=*BY-DSSM) ————————————————————  (4)
//SAVE-SSD ——————————————————————————————————————————————————————————  (5)
```

(1) Deklaration des SSD-Objects.

(2) Definition der wichtigsten Attribute des Subsystems SS1: der Auftragseingänge, der Referenzen und Abhängigkeiten.

(3) Bestimmung der Subsysteme, die keinen gemeinsamen Adressraum mit SS1 haben dürfen; im Beispiel die Subsysteme SS5, SS6 und SS9.

(4) Bestimmung der Holdertask-Eigenschaften von SS1, SS2 und SS3.

(5) Sichern des SSD-Objects, das die Definition des Subsystems SS1 enthält. Zusätzlich wird das SSD-Object in die bei (1) angegebene ISAM-Datei SYSSSC.SS1.001 abgespeichert.

4.5.2 Erstellung eines statischen Subsystemkatalogs

```
/START-SSCM
//START-CATALOG-CREATION CATALOG-NAME=KAT1 ————————————————————————————  (1)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.SS1.001,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————————  (2)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSD.SS2.001,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ————————  (3)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS3(VERSION=V2.5A00),
        :
        : ————————————————————————————————————————————————————————————————  (4)
//CHECK-CATALOG ————————————————————————————————————————————————————————————  (5)
//SAVE-CATALOG CATALOG-NAME=*CURRENT,FORCED=*NO ————————————————————————————  (6)
```

(1) Deklaration des statischen Subsystemkatalogs.

(2) Einbringen der Subsystemdefinition in der Object-Form für SS1 in den Katalog.

(3) Einbringen der Subsystemdefinition im UGEN-Format für SS2 in den Katalog.

(4) Definition der Attribute des Subsystems SS3.

(5) Prüfen des Katalogs.

(6) Sichern des erzeugten statischen Subsystemkatalogs.

4.5.3 Änderung eines statischen Subsystemkatalogs

```
/START-SSCM
//START-CATALOG-MODIFICATION CATALOG-NAME=KAT1 ————————————————————  (1)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.SS5,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ———  (2)
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSD.SS4,
        INSTALLATION-USERID=*UNCHANGED,CORRECTION-STATE=*UNCHANGED ———  (3)
//SET-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS1(V01.0A10),
        LINK-ENTRY=SS1ENT,MEMORY-CLASS=*SYSTEM-GLOBAL(SUBSYSTEM-ACCESS=
        *SYSTEM),INIT-ROUTINE=INITROUT,DEINIT-ROUTINE=*DYNAMIC,
        INTERFACE-VERSION=INTVERS ————————————————————————————————  (4)
//SEPARATE-ADDRESS-SPACE SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=(SS5,SS6) ————————————————————————————————  (5)
//ASSIGN-HOLDER-TASK TYPE=*WORK-TASK(SUBSYSTEM-NAME=SS1,
        SUBSYSTEM-VERSION=V01.0A10) ———————————————————————————————  (6)
//REMOVE-CATALOG-ENTRY SUBSYSTEM-NAME=SS7(VERSION=V03.2A00) ————————  (7)
//MODIFY-SUBSYSTEM-ATTRIBUTES SUBSYSTEM-NAME=SS2(VERSION=V11.0A10),
        INSTALLATION-USERID=UIDXYZ,....
        : ————————————————————————————————————————————————————————  (8)
//REMOVE-ADDR-SPACE-SEPARATION SUBSYSTEM-NAME=SS1,
        FROM-SUBSYSTEMS=SS2 ——————————————————————————————————————  (9)
//MODIFY-WORK-TASK-ATTRIBUTE SUBSYSTEM-NAME=SS3,
        SUBSYSTEM-VERSION=00.1,WORK-TASK=*NO  ————————————————————— (10)
//CHECK-CATALOG ————————————————————————————————————————————————— (11)
//SAVE-CATALOG CATALOG-NAME=*CURRENT,FORCED=*NO ————————————————— (12)
```

(1) Angabe des Katalogs, dessen Inhalt geändert werden soll.

(2) Hinzufügen einer Subsystemdefinition, die in einem SSD-Object enthalten ist.

(3) Hinzufügen einer Subsystemdefinition, die in der alten DSSMGEN-Syntax (UGEN-Format) geschrieben wurde.

(4) Hinzufügen eines neuen Subsystems unter Bestimmung seiner Attribute.
-(6)

(7) Löschen einer Subsystemdefinition aus dem Katalog.

(8) Ändern einer Subsystemdefinition, die sich bereits im Katalog befindet.

(9) Aufheben der strikten Adressraumteilung für beide Subsysteme.

(10) Ändern der Arbeitstask-Attribute eines als Arbeitstask arbeitenden Subsystems.

(11) Prüfen des Katalogs.

(12) Sichern des geänderten statischen Subsystemkatalogs.

5 Literatur

Die Handbücher finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com.

ADAM (BS2000)
Abstract Device Access Method
Benutzerhandbuch

AID (BS2000)
Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

ARCHIVE (BS2000)
Benutzerhandbuch

BLSSERV
Bindelader-Starter
Benutzerhandbuch

BINDER (BS2000)
Benutzerhandbuch

CALENDAR (BS2000)
Benutzerhandbuch

CRTE (BS2000)
Common RunTime Environment
Benutzerhandbuch

DAB (BS2000)
Disk Access Buffer
Benutzerhandbuch

BS2000 OSD/BC
Dienstprogramme
Benutzerhandbuch

Distributed Print Services (BS2000)
Drucken in Computernetzen
Benutzerhandbuch

DRV (BS2000)
Dual Recording by Volume
Benutzerhandbuch

EDT (BS2000)
Anweisungen
Benutzerhandbuch

BS2000 OSD/BC
Einführung in das DVS
Benutzerhandbuch

BS2000 OSD/BC
Einführung in die Systembetreuung
Benutzerhandbuch

BS2000 OSD/BC
Systeminstallation
Benutzerhandbuch

FDDRL (BS2000)
Benutzerhandbuch

HSMS (BS2000)
Hierarchisches Speicher Management System
2 Bände
Benutzerhandbücher

Band 1 enthält die Beschreibung der Funktionen, der Verwaltung und der Installation
Band 2 enthält die Beschreibung der HSMS-Anweisungen in alphabetischer Reihenfolge

IMON (BS2000)
Installationsmonitor
Benutzerhandbuch

JV (BS2000)
Jobvariablen
Benutzerhandbuch

BS2000 OSD/BC
Kommandos (Band 1-7)
Benutzerhandbuch

LMS (BS2000)
SDF-Format
Benutzerhandbuch

MAREN (BS2000)
Band 1 und 2
Benutzerhandbücher

Band 1 enthält eine Einführung in die Arbeit mit MAREN.
Band 2 enthält Übersichten, Schnittstellenbeschreibungen und Beispiele zur Arbeit mit MAREN, aufgeteilt in einen privilegierten und einen nichtprivilegierten Teil.

HIPLEX MSCF (BS2000)
BS2000-Rechner im Verbund
Benutzerhandbuch

PCS (BS2000)
Performance Control Subsystem
Benutzerhandbuch

POSIX (BS2000)
Grundlagen für Anwender und Systemverwalter
Benutzerhandbuch

POSIX (BS2000)
Kommandos
Benutzerhandbuch

PROP-XT (BS2000)
Programmiertes Operating mit komfortablen Sprachmitteln von SDF-P
Produkthandbuch

RFA (BS2000)
Remote File Access
Benutzerhandbuch

RSO (BS2000)
Remote SPOOL Output
Benutzerhandbuch

SDF-A (BS2000)
Benutzerhandbuch

SDF-P (BS2000)
Programmieren in der Kommandosprache
Benutzerhandbuch

SDF (BS2000)
SDF-Verwaltung
Benutzerhandbuch

SECOS (BS2000)
Security Control System
Benutzerhandbücher

openSM2 (BS2000)
Software Monitor
Benutzerhandbuch

SPOOL (BS2000)
Benutzerhandbuch