z/OS Job Control Language

Schulungsunterlagen

26. Januar 2015

Eine Ausarbeitung von:

cps4it

Ralf Seidler • Stromberger Straße 36A • 55411 Bingen

Fon: +49-6721-992611 • Fax: -49-6721-992613 • Mail: ralf.seidler@cps4it.de

Internet : http://www.cps4it.de

Steuernummer: 08/220/2497/3, Finanzamt Bingen, Ust-ID : DE214792185

Inhaltsverzeichnis

1. VORBEREITUNGEN FEHLER! TEXTMARKE NICHT DEFINIERT.

1.1 ANMELDEN UND TEST DER USER-IDEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
1.2 BIBLIOTHEK FÜR ÜBUNGSAUFGABEN ERSTELLEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**

2. JOBKARTE FEHLER! TEXTMARKE NICHT DEFINIERT.

2.1 ERSTELLEN EINER JOBANWEISUNG **FEHLER! TEXTMARKE NICHT DEFINIERT.**
2.2 JOBKARTE ERWEITERN - 1 **FEHLER! TEXTMARKE NICHT DEFINIERT.**
2.3 JOBKARTE ERWEITERN - 2 **FEHLER! TEXTMARKE NICHT DEFINIERT.**
2.4 JOBSTEP ERSTELLEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
2.5 JOB MIT 2 STEPS **FEHLER! TEXTMARKE NICHT DEFINIERT.**
2.6 JOB MIT COND-STEUERUNG **FEHLER! TEXTMARKE NICHT DEFINIERT.**

3. ARBEITEN MIT DATEIEN – 1 FEHLER! TEXTMARKE NICHT DEFINIERT.

3.1 LESEN EINER INSTREAM-DATEI **FEHLER! TEXTMARKE NICHT DEFINIERT.**
3.2 LESEN EINER DUMMY-DATEI **FEHLER! TEXTMARKE NICHT DEFINIERT.**
3.3 SCHREIBEN IN DAS „NIRWANA“ **FEHLER! TEXTMARKE NICHT DEFINIERT.**

4. ARBEITEN MIT DATEIEN – 2 FEHLER! TEXTMARKE NICHT DEFINIERT.

4.1 ANLEGEN EINER SEQUENTIELLEN DATEI **FEHLER! TEXTMARKE NICHT DEFINIERT.**
4.2 ANLEGEN EINER PO-DATEI **FEHLER! TEXTMARKE NICHT DEFINIERT.**
4.3 KOPIEREN IN EIN MEMBER EINER PO-DATEI **FEHLER! TEXTMARKE NICHT DEFINIERT.**
4.4 KOPIEREN PO-MEMBER **FEHLER! TEXTMARKE NICHT DEFINIERT.**

5. UTILITIES FEHLER! TEXTMARKE NICHT DEFINIERT.

6. JOBSTEUERUNG, STEPSTEUERUNG FEHLER! TEXTMARKE NICHT DEFINIERT.

6.1 ANLEGEN EINER SEQUENTIELLEN DATEI MIT RÜCKBEZUG **FEHLER! TEXTMARKE NICHT DEFINIERT.**
6.2 KOPIEREN EINER DATEI MIT EVENTUELLER NEUANLAGE **FEHLER! TEXTMARKE NICHT DEFINIERT.**

7. JOBS MIT LADEBIBLIOTHEKEN FEHLER! TEXTMARKE NICHT DEFINIERT.

7.1 PO-DATEI ALS LADEBIBLIOTHEK ANLEGEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.2 JOB MIT STEPLIB **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.3 JOB MIT JOBLIB **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.4 JOB MIT FALSCHEM PROGRAMMNAMEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.5 ANLEGEN GDG-BASE-ENTRY **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.6 KOPIEREN DATEN AUF GDS **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.7 ARBEITEN MIT AKTUELLEM GDS **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.8 ARBEITEN MIT ALLEN GENERATIONEN EINER GDG **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.9 LÖSCHEN GDG **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.10 STORCLAS IM UNTERNEHMEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
7.11 MGMTCLAS IM UNTERNEHMEN **FEHLER! TEXTMARKE NICHT DEFINIERT.**

8. JOBS MIT PROZEDUREN FEHLER! TEXTMARKE NICHT DEFINIERT.

8.1 PROZEDURAUFRUF - INSTREAM **FEHLER! TEXTMARKE NICHT DEFINIERT.**
8.2 PROZEDURAUFRUF – INSTREAM, PARAMETRISIERT **FEHLER! TEXTMARKE NICHT DEFINIERT.**
8.3 PROZEDURAUFRUF – EXTERN **FEHLER! TEXTMARKE NICHT DEFINIERT.**
8.4 PROZEDURAUFRUF – EXTERN, PARAMETRISIERT **FEHLER! TEXTMARKE NICHT DEFINIERT.**
8.5 PROZEDURAUFRUF – MSGLEVEL **FEHLER! TEXTMARKE NICHT DEFINIERT.**

Inhalt siehe http://wiki.cps4it.de

1.

Einführung

1.1 Betriebsystem z/OS und Job Control Language

1.1.1 Begriffsdefinitionen

Job, Step

Letztendlicher Zweck eines Betriebssystems ist es, die Abarbeitung von Programmen und die Verarbeitung von Daten zu ermöglichen.

Unter einem Arbeitsauftrag (Job) versteht man einen Auftrag an das Betriebssystem, eine oder mehrere vordefinierte Arbeitsschritte (Steps) mit jeweils einem Programm unter Benutzung einer oder mehrerer Dateien abzuarbeiten.

Wird dieser Auftrag schriftlich und vollständig unter Benutzung einer formalisierten Sprache, der Job Control Language (JCL) formuliert, spricht man von einem Stapelverarbeitungs-Job (Batch-Job) in Anlehnung an die ursprüngliche Form als Stapel von Lochkarten.

Der Auftrag besteht aus einzelnen JCL-Anweisungen, in denen die für die Abarbeitung des Auftrages notwendigen Informationen enthalten sind.

Die wichtigsten JCL-Anweisungen sind

- Die Job-Anweisung, die den Beginn eines Jobs kennzeichnet
- Die EXEC-Anweisung, die den Beginn eines Steps kennzeichnet. Ein Job kann aus einem oder mehreren Steps bestehen.
- Die DD-Anweisung ,die eine zu verarbeitende Datei beschreibt. In einem Step können mehrere Dateien spezifiziert werden.

Job Entry Subsystem

Im MVS/ESA ist das Job Entry Subsystem (JES) zuständig für die Abarbeitung von Batch-Jobs. Das JES

- liest die mit Hilfe der JCL formulierten Arbeitsaufträge und Arbeitsschritte ein und interpretiert sie
- bereitet deren Ausführung vor und überwacht die Ausführung
- verwaltet das Jobablauf-Protokoll sowie etwaige Druckausgaben und steuert deren tatsächliche Ausgabe auf Drucker

Für MVS/ESA existieren zwei JES-Systeme, JES2 und JES3. Beide Systeme beinhalten dieselben Grundfunktionen, in Details unterscheiden sie sich jedoch teilweise erheblich.

1.1.2 JES2, JES3

JES2

JES2 wurde ursprünglich als Job-Steuersystem für Einzelrechner entwickelt. Das bedeutet, dass ein Job auf *einem* Rechner eingelesen und ausgeführt wird, und das Jobablauf-Protokoll sowie etwaige Druckausgaben auf *demselben* Rechner verwaltet werden.

Später wurden Funktionen hinzugefügt, die eine gemeinsame Verwaltung von Jobs durch mehrere Einzelrechner mit JES2-Systemen ermöglichen.

Einzelne Jobs laufen grundsätzlich unabhängig voneinander.

JES3

JES3 wurde als Jobsteuerungssystem für mehrere gekoppelte Rechner mit MVS-Systemen entwickelt. Das bedeutet, dass ein Job auf *einem* Rechner eingelesen, auf einem *anderen* Rechner ausgeführt, und das Jobablauf-Protokoll sowie etwaige Druckausgaben auf wieder einem *anderen* Rechner verwaltet werden kann.

Außerdem ist es in JES3 möglich, Abhängigkeiten zwischen einzelnen Jobs in Form von so genannten Job-Netzen zu definieren.

Gemeinsamkeiten

Die JCL für JES2 und JES3 ist weitgehend gleich. Für die nachfolgenden Erläuterungen sind die Unterschiede meist vernachlässigbar.

Die Detailunterschiede haben jedoch gewisse Einflüsse auf die Technik, wie man Jobs optimal angepasst an JES2 bzw. JES3 gestalten sollte.

Spool-Datei

Verwaltungsinformationen werden über die

- eingelesenen und noch zu verarbeitenden Jobs
- in Verarbeitung befindlichen Jobs
- bereits verarbeiteten, noch nicht „gelöschten“ Jobs

sowie

- etwaige, zu diesen Jobs gehörende Eingabedaten
- Jobablauf-Protokolle
- etwaige, zu diesen Jobs gehörende Druckausgaben, die noch nicht gedruckt, weitergeleitet oder gelöscht wurden

in speziellen Dateien auf Platte, so genannten Spool-Dateien gespeichert. Diese Spool-Dateien stehen unter direkter Verwaltung durch das JES.

1.2 Phasen der Jobverarbeitung

1.2.1 Überblick

Erstellen der JCL-Anweisungen

Das Konzept der JCL stammt aus der Lochkartenzeit. Für jede JCL-Anweisung wurde jeweils eine 80-stellige Karte gestanzt.
Heute wird nicht mehr mit Lochkarten gearbeitet. Die Codier- und Eingabegeräte wurden durch Terminals (Bildschirme und Tastatur) ersetzt. Jede Zeile, die eingegeben wird, entspricht einer Lochkarte. Weil eine Lochkarte 80 Stellen hatte, haben auch heute noch die JCL-Anweisungen und viele Dateien eine logische Satzlänge von 80 Bytes.

Einlesen der JCL-Anweisungen

Ursprünglich wurden die auf Lochkarten gestanzten Jobs bzw. die JCL-Anweisungen mit Hilfe eines Lochkarten-Lesers eingelesen.
Heute werden die Jobs bzw. die JCL-Anweisungen mit Hilfe eines Dialogsystems (z.B. TSO oder ISPF) erfasst und mit dem TSO-Kommando „SUBMIT“ an das JES weitergeleitet. Dieser Vorgang wird als „Submit“ bezeichnet.
In beiden Fällen trennt das JES die JCL-Anweisungen von Daten, die innerhalb der JCL-Anweisungen eingebettet sind. Diese Daten werden als Instream-Daten bezeichnet. JCL-Anweisungen und Instream-Daten werden getrennt in der Spool-Datei gespeichert.

Vorbereiten des Joblaufes

JCL-Anweisungen sind in der Form, wie sie vom Menschen erstellt werden, nicht vom System verarbeitbar. Sie werden daher nach dem Einlesen in eine interne Form umgesetzt. Dieser Vorgang wird als „Konvertieren“ bezeichnet.

Dabei findet auch eine Überprüfung der JCL auf syntaktische Richtigkeit statt. Formelle Fehler werden erkannt. Jobs mit solchen Fehlern werden nicht zur Verarbeitung angenommen. In das Jobablauf-Protokoll wird eine entsprechende Fehlermeldung aufgenommen.

Die konvertierte JCL von syntaktisch fehlerfreien Jobs wird ebenfalls in der Spool-Datei abgelegt.
JES3 führt vorbereitend auch ein „Interpretieren“ der JCL-Anweisungen (Scheduling) durch. Dabei wird z.B. geprüft, ob die zur Ausführung des gesamten Jobs benötigten Geräte, Datenträger und Datenbestände verfügbar sind. Bestimmte Fehlerfälle, z.B. der Versuch, einen nicht existierenden Datenbestand verarbeiten zu wollen, wird vom JES3 bereits jetzt erkannt und mit einer Fehlermeldung abgewiesen.

Ausführen des Joblaufes

Bevor ein Job ausgeführt werden kann, müssen die beschriebenen Vorbereitungsmaßnahmen erledigt und sowie einige Bedingungen erfüllt sein.

Hauptbedingung ist, dass wenigstens ein MVS/ESA-Adressraum, der für Batch-Jobs reserviert ist, auch tatsächlich verfügbar und frei ist. Wie viele MVS/ESA-Adressräume hierfür zur Verfügung stehen (einige wenige bis zu vielen hundert), ist installationsabhängig.

In einem JES2-System geschieht erst bei Beginn der Jobausführung das „Interpretieren“ der JCL-Anweisungen und die Zuordnung der für die Ausführung des ersten Steps benötigten Geräte, Datenträger und Datenbestände. Eventuelle Fehlersituationen fallen erst jetzt auf. (Die Zuordnung für die späteren Steps dieses Jobs erfolgt in einem JES2-System beim Beginn des jeweiligen Steps).

Bei Stepende werden diejenigen Geräte, Datenträger und Datenbestände freigegeben, die in den nachfolgenden Steps nicht mehr benötigt werden. Bei Jobende schließlich werden alle noch zugeordneten Geräte, Datenträger und Datenbestände freigegeben.

Während des Joblaufes wird ein Jobablauf-Protokoll erstellt, das in der Spool-Datei abgelegt wird. Auch evtl. entstehende Druckausgaben werden in der Spool-Datei abgelegt.

Ausgabe, Löschen

Die Ausgabe des Jobablauf-Protokolls und evtl. Druckausgaben erfolgten früher grundsätzlich auf einen Drucker. Heute sind verschiede Verfahren der papierlosen Weiterverarbeitung oder Archivierung bekannt, z.B. Mikroverfilmung, Speicherung auf CD, usw. In manchen Fällen, z.B. beim Testen, genügt es, den Inhalt dieser Protokolle und Druckausgaben am Bildschirm zu betrachten.

Die Ausgabe kann somit

- durch JES gesteuert auf Drucker erfolgen. Danach werden die in der Spool-Datei zwischengespeicherten Daten automatisch gelöscht.
- in Form einer Weiterverarbeitung anderer Art geschehen oder durch ein Betrachten am Bildschirm ersetzt werden. Nicht mehr benötigte, zwischengespeicherte Daten müssen manuell aus der Spool-Datei gelöscht werden.

Schließlich können alle zu einem Job gehörenden Verwaltungsinformationen, Jobablauf-Protokolle und Druckausgaben insgesamt gelöscht werden [Purge]. Damit sind alle Spuren dieses Jobs getilgt.

1.3 Eingabe-, Ausgabe-Warteschlangen

1.3.1 Überblick

JES benutzt zur Verwaltung der Jobs

- Jobnamen
- Jobnummern
- Jobklassen und Eingabe-Warteschlangen
- Ausgabeklassen und Ausgabe-Warteschlangen

1.3.2 Eingabe

Jobname

Der Jobname (Name eines Jobs) wird durch den Ersteller des Jobs in der JOB-Anweisung festgelegt. Jobnamen können mehrfach verwendet werden. Sie sind zwar ein Identifizierungsmerkmal für einen Job, aber nicht notwendigerweise eindeutig.

Jobnummer

Beim Einlesen eines Jobs ordnet JES jedem Job eine eindeutige Jobnummer (in der Form JOBnnnnn) zu. Die Jobnummer ist somit ein eindeutiges Identifizierungsmerkmal für einen Job.

Jobklasse, Eingabe-Warteschlange

JES verwaltet die eingelesenen, aber noch nicht in Ausführung befindlichen Jobs in 36 Eingabe-Warteschlangen mit den Bezeichnungen A. . .Z und 0.. .9. In welche Warteschlange ein Job eingereiht wird, wird durch den Ersteller des Jobs in der JOB Anweisung mit Hilfe des CLASS-Parameters festgelegt. Die Angabe einer Jobklasse legt somit die zu verwendende Eingabe-Warteschlange fest.
Normalerweise werden die Jobs in derjenigen Reihenfolge in die zuständige Warteschlange gestellt, in der sie eingelesen werden. Es ist jedoch mit Hilfe einer Prioritäten-Definition möglich, die Reihenfolge nach Belieben zu gestalten.
Außerdem kann der Systemprogrammierer an eine bestimmte Jobklasse verschiedene Eigenschaften knüpfen, beispielsweise ein CPU-Zeitlimit oder die Verwendbarkeit bestimmter Geräte (z. B. Magnetbandgeräte).
Zusätzlich existiert eine HOLD Warteschlange. Jobs in dieser Warteschlange sind nicht zur Abarbeitung freigegeben. Sie können jedoch zu einem späteren Zeitpunkt vom Ersteller des Jobs oder vom Operating in eine der „normalen“ Eingabe-Warteschlangen umgesetzt werden.

1.3.3 Ausführung

„Initiator“-Programm

Das Zur-Verfügung-Stellen eines Adressraumes zur Jobausführung geschieht dadurch, dass ein bestimmtes Programm, das „Initiator-Programm, in einem Adressraum läuft und aus der Menge der anstehenden Jobs nach bestimmten Kriterien einen Job zur Ausführung in diesem Adressraum auswählt.

Ein „Initiator-Programm, das in einem Adressraum läuft, durchsucht eine oder mehrere bestimmte Eingabe-Warteschlangen, um den nächsten auszuführenden Job auszuwählen.

Wie viele „Initiator“-Programme gleichzeitig aktiv sind und für welche Eingabe-Warteschlange(n) ein bestimmtes „Initiator-Programm zuständig ist, legt der Systemprogrammierer fest bzw. kann vom Operating, z.B. abhängig von der Tageszeit, verändert werden.

Gibt es nur ein einziges „Initiator-Programm, das eine bestimmte Eingabe-Warteschlange (Jobklasse) abarbeitet, so wird dadurch eine Serialisierung von Jobs mit dieser Jobklasse erzwungen.

Andererseits kann durch die Tatsache, dass viele „Initiator-Programme dieselbe Eingabe-Warteschlange (Jobklasse) abarbeiten, eine intensive Parallelverarbeitung bestimmter Jobs erreicht werden.

Jobs in Eingabe-Warteschlangen, für die kein „Initiator-Programm „zuständig“ ist, werden nicht abgearbeitet.

Bedeutung des Jobnamens

Ein Job wird solange nicht zur Ausführung ausgewählt, solange sich ein anderer Job mit demselben Jobnamen in der Ausführungsphase befindet. Damit besteht für den Ersteller von Jobs eine Möglichkeit, die Serialisierung mehrerer Jobs zu erzwingen.

1.3.4 Ausgabe

Ausgabeklasse, Ausgabe-Warteschlange

JES verwaltet die Jobablauf-Protokolle und Druckausgaben in 36 Ausgabe-Warteschlangen mit den Bezeichnungen A. . .Z und 0... 9.

In welche Warteschlange ein Jobablauf-Protokoll eingereiht wird, wird durch den Ersteller des Jobs in der JOB-Anweisung mit Hilfe des MSGCLASS Parameters festgelegt.

In welche Warteschlange eine Druckausgabe eingereiht wird, wird durch den Ersteller des Jobs in der DD-Anweisung mit Hilfe des SYSOUT Parameters festgelegt.

Die Angabe einer Ausgabeklasse legt somit die zu verwendende Ausgabe-Warteschlange fest. Jobablauf-Protokoll und Druckausgaben können in dieselbe oder in unterschiedliche Warteschlangen gestellt werden.

Zusätzlich existiert eine HOLD Warteschlange. Objekte in dieser Warteschlange sind nicht zur Ausgabe freigegeben Sie können jedoch zu einem späteren Zeitpunkt vom Ersteller des Jobs oder vom Operating in eine der „normalen“ Ausgabe-Warteschlangen umgesetzt werden.

Druckausgabe

Ein Drucker wird von einem JES spezifischen Druckausgabe-Programm bedient. Dieses Programm durchsucht eine oder mehrere bestimmte Ausgabe-Warteschlangen, um das nächste auszugebende Jobablauf-Protokoll bzw. die nächste Druckausgabe auszuwählen.

Es können beliebig viele Drucker und somit Druckausgabe-Programme gleichzeitig aktiv sein. Für welche Ausgabe-Warteschlange(n) ein bestimmter Drucker „zuständig“ ist, legt der Systemprogrammierer fest bzw. kann vom Operating, z.B. abhängig von der Art des zu bedruckenden Papiers, verändert werden.

Objekte in Ausgabe-Warteschlangen, für die kein Drucker „zuständig“ ist, werden nicht abgearbeitet. Dieser Effekt kann beabsichtigt sein, wenn die Objekte nie gedruckt, sondern nur am Bildschirm betrachtet werden sollen.

1.4 JCL-Anweisungen

1.4.1 Überblick

JOB-Anweisung

- ist immer die erste Anweisung in einem Job. Legt den Jobnamen fest
- beschreibt einen Job und dessen Eigenschaften

EXEC-Anweisung

- identifiziert den Beginn eines Steps Beschreibt dessen Eigenschaften
- gibt das auszuführenden Programm bzw. die JCL Prozedur an

DD-Anweisung

- beschreibt die Eigenschaften eines Datenbestandes

Kommentar-Anweisung

- enthält eine Kommentarzeile, d.h. einen beliebigen Text

Delimiter-Anweisung

- kennzeichnet das Ende von Instream Daten, d.h. von Daten, die zwischen den JCL-Anweisungen eingebettet sind

IF/THEN-, ELSE-, ENDIF-Anweisungen

- IF gibt eine Bedingung an, unter der die im THEN-Zweig angegebenen Steps ausgeführt werden sollen
- andernfalls werden die im ELSE-Zweig angegebenen Steps ausgeführt
- ENDIF kennzeichnet das Ende der IF/THEN/ELSE-Struktur

JCLLIB-Anweisung

- identifiziert eine oder mehrere JCL-Bibliotheken, in denen INCLUDE-Elemente und/oder JCL-Prozeduren gesucht werden

INCLUDE-Anweisung

- gibt ein Element einer JCL-Bibliothek an, das vordefinierte JCL-Anweisungen enthält. Diese JCL-Anweisungen werden dem Job hinzugefügt.

PROC-Anweisung

- definiert den Beginn einer JCL-Prozedur. Sie gibt den Prozedurnamen an.

PEND-Anweisung

- definiert das Ende einer JCL-Prozedur

SET-Anweisung

- weist symbolischen Parametern aktuelle Werte zu

OUTPUT-Anweisung

- legt Einzelheiten der Druckausgabe fest

1.4.2 Syntax

Allgemeines

Eine JCL-Anweisung besteht physisch aus einem oder mehreren Sätzen (Zeilen) zu je 80 Zeichen (Bytes). Jede JCL-Anweisung ist aus mehreren Teilbereichen aufgebaut.

Identifikationsfeld

In den Spalten 1 und 2 einer JCL-Anweisung steht die Zeichenfolge '// (Ausnahme Delimiter-Kennzeichnung /*),

Namensfeld

Ab Spalte 3 einer JCL-Anweisung steht ein Namensfeld. Je nach Art der JCL-Anweisung kann oder muss ein Name angegeben werden. Der Name darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A.. .Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A.. .Z, 0.. .9) oder eines der Sonderzeichen § # $ sein. Dem Namen muss mindestens eine Leerstelle folgen.

Operationsfeld

Hier steht der Operationscode, d.h. die Bezeichnung der JCL-Anweisung. Diese Angabe ist immer erforderlich (z.B. JOB, EXEC, DD, usw.) Dem Operationscode muss mindestens eine Leerstelle folgen.

Parameterfeld

Das Parameterfeld enthält Positions- und/oder Schlüsselwortparameter. Mehrere Parameter werden durch Kommata voneinander getrennt.
Das Parameterfeld wird beim Auftreten einer Leerstelle als beendet betrachtet. Spätestens endet es in Spalte 71. Innerhalb des Parameterfeldes darf keine Leerstelle vorhanden sein.

Kommentarfeld

Ein Kommentar ist eine beliebige (optionale) Zeichenfolge, die nur zur Information des Lesers dient, aber vom System ignoriert wird. Das Kommentarfeld endet spätestens in Spalte 71. Zwischen Parameterfeld und Kommentarfeld muss mindestens eine Leerstelle vorhanden sein.

Fortsetzungsfeld

In Spalte 72 kann ein beliebiges Fortsetzungskennzeichen (irgendein Zeichen, ausgenommen Leerstelle) angegeben werden, falls eine JCL-Anweisung nicht in einer Zeile untergebracht werden kann und in der nachfolgenden Zeile fortgesetzt werden soll. Soll keine Fortsetzung per Folgezeile erfolgen, dann muss Spalte 72 leer bleiben. Soll tatsächlich eine Folgezeile benutzt werden, so wird die ursprüngliche Zeile innerhalb des Parameterfeldes unmittelbar nach einem beliebigen Komma unterbrochen.

Folgezeile

Die Folgezeile muss in den Spalten 1 und 2 die Zeichenfolge ‘// enthalten, gefolgt von einer Leerstelle in Spalte 3. Die fortzusetzenden Parameter an gaben dürfen frühestens in Spalte 4 und müssen spätestens in Spalte 16 beginnen. Es können auch mehrfache Folgezeilen benutzt werden.

Nummerierungsfeld

In den Spalten 73 bis 80 aller JCL-Anweisungen kann eine Nummerierung der einzelnen Anweisungen untergebracht werden. Diese Nummerierung ist jedoch in der Regel bedeutungslos.

1.4.3 Parameterarten

Allgemeines

Parameter sind Operanden für eine bestimmte JCL-Anweisung. Sie werden abhängig von ihrer äußeren Form in Positionsparameter und Schlüsselwortparameter unterteilt.

Positionsparameter

Positionsparameter sind durch ihre Position (d.h. durch die Reihenfolge) im Parameterfeld gekennzeichnet. Die vorgegebene Reihenfolge muss eingehalten werden. Das Weglassen eines Positionsparameters muss, falls weitere Positionsparameter folgen, durch ein Komma verdeutlicht werden. Alle Positionsparameter müssen vor allen Schlüsselwortparametern angegeben werden.

Schlüsselwortparameter

Ein Schlüsselwortparameter besteht aus einem Schlüsselwort, dem ein Gleichheitszeichen und dahinter ein aktueller Wert folgt. Die Reihenfolge der Schlüsselwortparameter im Parameterfeld ist beliebig. Sie müssen aber den Positionsparametern folgen.

Subparameter

Jeder Positions- oder Schlüsselwortparameter kann, falls erforderlich, in Subparameter unterteilt sein. Die Subparameter werden ebenfalls durch ein Komma voneinander getrennt und durch Klammerung zu einem Parameter zusammengefasst.

Hinweis

Beim Weglassen von Parametern gelten in vielen Fällen Standardwerte (Default-Werte), die in der Regel der Systemprogrammierer festgelegt hat.

2. Job-, Step-Beschreibung

2.1 JOB-Anweisung

2.1.1 Uberblick

Verwendung

Die JOB-Anweisung definiert den Beginn eines Jobs und beschreibt dessen Verarbeitungsmerkmale. Die JOB-Anweisung ist immer die erste Anweisung in einem Job. Innerhalb eines lnput-Streams bedeutet eine nachfolgende JOB-Anweisung das Ende des vorhergehenden Jobs.

Namensfeld

Das Namensfeld enthält den Jobnamen. Der Jobname ist prinzipiell frei wählbar. In nahezu allen MVS-Systemen gibt es jedoch installationsspezifische Vorschriften für die Wahl des Jobnamens.
Spalte 1 und 2 enthalten '//'. Der Jobname muss in Spalte 3 beginnen. Er darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A. ..Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0.. .9) oder eines der Sonderzeichen § # $ sein. Dem Jobnamen muss mindestens eine Leerstelle folgen.

Operationsfeld

Der Operationscode muss JOB lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt. Dem Operationscode muss mindestens eine Leerstelle folgen.

Parameterfeld

Das Parameterfeld enthält Positions- und Schlüsselwortparameter. Alle Parameter sind wahlweise (installationsabhängig). Die Positionsparameter sind, falls vorhanden, zuerst zu schreiben. Die Schlüsselwortparameter folgen den Positionsparametern.
Der Systemprogrammierer kann Standardwerte (Default-Werte) vorgeben. Ein Standardwert wird dann wirksam, wenn ein bestimmter Parameter nicht angegeben wird.

2.1.2 Positionsparameter: Abrechnungs-Information

Verwendung

Die Abrechnungs-Information (der Accounting-Parameter) dient der Zuordnung des Jobs zu einer bei der Installation definierten Kostenstelle. Diese Informationen werden zur späteren Auswertung für Abrechnungszwecke in einer Datei zusammen mit Kostenfaktoren dieses Jobs gespeichert.
Die Angabe dieses Parameters ist optional, kann jedoch durch den Systemprogrammierer zur Pflicht gemacht werden.

Parametertyp

Erster Positionsparameter, optional (installationsabhängig).

Syntax

Für die Abrechnungs-Informationen [accounting information] sind max. 143 Zeichen erlaubt.
Werden mehrere (maximal 4) Subparameter als Abrechnungsinformationen verwendet, so müssen diese durch ein Komma voneinander getrennt und durch Klammerung oder durch Hochkommata zu einem Gesamt-Parameter zusammengefasst werden.

Sonderzeichen

Bei der Verwendung von Sonderzeichen ist die entsprechende Zeichenfolge (gesamter Parameter oder der jeweilige Subparameter) in Hochkommata zu setzen.
Die Sonderzeichen ’ und & müssen grundsätzlich verdoppelt werden. Der Bindestrich gilt nicht als Sonderzeichen.

Hinweis

Wird dieser Parameter weggelassen, aber der zweite Positionsparameter (Programmer‘s Name) verwendet, so muss das Fehlen des ersten Positionsparameters durch ein Komma angezeigt werden

2.1.3 Positionsparameter: Programmierername

Verwendung

Der Programmierername (Programmer‘s Name-Parameter) dient der Identifizierung und Zuordnung eines Jobs zu einer Person bzw. Personengruppe.
Die Angabe dieses Parameters ist optional, kann jedoch durch den Systemprogrammierer zur Pflicht gemacht werden.

Parametertyp

Zweiter Positionsparameter, optional (installationsabhängig).

Syntax

Für den Programmierernamen (programmer‘s name) sind max. 20 Zeichen erlaubt.

Sonderzeichen

Bei der Verwendung von Sonderzeichen ist die gesamte Zeichenfolge in Hochkommata zu setzen. Die Sonderzeichen ‘ und & müssen grundsätzlich verdoppelt werden. Der Bindestrich gilt nicht als Sonderzeichen.

Hinweis

Wird dieser Parameter weggelassen, so wird durch ein Komma das Fehlen dieses Parameters angezeigt. Fehlen beide Positionsparameter, dann reicht ein Komma aus. Man kann auch zwei Kommas angeben, aber im Protokoll wird dann nur ein Komma angezeigt.

2.1.4 Schlüsselwortparameter: TYPRUN

Verwendung

Der TYPRUN-Parameter steuert spezielle Verfahren zur Job-Behandlung durch JES2/3.
Es ist möglich:

- die JCL-Anweisungen lediglich einer Syntaxprüfung zu unterziehen.
- den Job in den HOLD-Status zu setzen.

Parametertyp

Schlüsselwortparameter, optional

TYPRUN=SCAN

Der Job wird einer JCL-Syntaxprüfung (ohne Ausführung) unterzogen.

TYPRUN=HOLD

Der Job wird vor der Ausführung in der HOLD-Warteschlange zurückgehalten, bis er vom Operating mit Hilfe eines Operator-Kommandos freigegeben wird.

2.1.5 Schlüsselwortparameter: CLASS

Verwendung

Der CLASS-Parameter ermöglicht die Zuordnung einer Jobklasse, unter der ein Job in die Job Input Queue (Eingabe-Warteschlange im JES2/3) eingeordnet werden soll.
Eine Jobklasse legt fest, ob und wie ein Job verarbeitet werden soll. In jeder Installation definiert der Systemprogrammierer die Eigenschaften jeder Jobklasse.

Parametertyp

Schlüsselwortparameter, optional

CLASS=jobclass

Gültige Jobklassen sind A-Z und 0-9. Die Eigenschaften einer bestimmten Klasse müssen in jeder Installation dokumentiert sein.

Standardannahme

Der Systemprogrammierer legt fest, welche Standard-Jobklasse (Default) bei Weglassen dieses Parameters gilt. Oft gilt dann CLASS=A.

2.1.6 Schlüsselwortparameter: MSGCLASS

Verwendung

Der MSGCLASS-Parameter ermöglicht die Zuordnung des Jobablauf-Protokolls zu einer Job Output Queue (Ausgabe-Warteschlange im JES2/3). Der Umfang des Jobablauf-Protokolls ist vom MSGLEVEL-Parameter abhängig.
Eine Ausgabeklasse legt fest, ob und wie die Systemausgaben verarbeitet werden sollen. In jeder Installation definiert der Systemprogrammierer die Eigenschaften jeder Ausgabeklasse.

Parametertyp

Schlüsselwortparameter, optional

MSGCLASS=class

Gültige Ausgabeklassen sind A-Z und 0-9. Die Eigenschaften einer bestimmten Klasse müssen in jeder Installation dokumentiert sein.

Standardannahme

Der Systemprogrammierer legt fest, welche Standard-Ausgabeklasse (Default) bei Weglassen dieses Parameters gilt.

Beziehungen zu anderen Parametern

MSGLEVEL-Parameter

2.1.7 Schlüsselwortparameter: MSGLEVEL

Verwendung

Der MSGLEVEL-Parameter bestimmt den Umfang des Jobablauf-Protokolls [Job Log]. Das Jobablauf-Protokoll ist die Zusammenstellung aller Job bezogenen Informationen.

Parametertyp

Schlüsselwortparameter, optional

MSGLEVEL=([statements] [,messages])

Subparameter: statements

Gibt an, in welchem Umfang JCL-Anweisungen im Jobablauf-Protokoll ausgegeben werden sollen:
0 nur JOB-Anweisung
1 alle JCL-, JES2/3-, und Prozedur-Anweisungen
2 nur JCL- und JES2/3-Anweisungen

Subparameter: messages

Gibt an, in welchem Umfang Systemnachrichten im Jobablauf-Protokoll ausgegeben werden sollen:
0 nur JCL-Nachrichten
Ausnahme: nach einem Job-Abbruch Umfang wie bei ‘1’
1 alle JCL-, JES- und Operator-Nachrichten

Standardannahme

Der Systemprogrammierer legt fest, welcher Standardwert (Default) bei Weglassen dieses Parameters gilt. Oft gilt dann MSGLEVEL=(1,1).

Beziehungen zu anderen Parametern

MSGCLASS-Parameter

Quellen für Systemnachrichten

IEF Job Scheduler
IEA Supervisor
IEC Data Management
ICH RACF

2.1.8 Schlüsselwortparameter: TIME

Verwendung

Der TIME-Parameter ermöglicht die Angabe eines Zeitlimits für den CPU-Zeitbedarf dieses Jobs.

Parametertyp

Schlüsselwortparameter, optional

TIME=([minuten] [,sekunden])

Gibt an, wie viele Minuten (1…357912) und/oder Sekunden (1…59) an CPU-Zeit dieser Job maximal benutzen darf. Nach Überschreiten dieser Zeit erfolgt ein Abbruch des Jobs (Abbruch-Code 322).

TIME=MAXIMUM

Gibt an, dass dieser Job 357912 Minuten an CPU-Zeit maximal benutzen darf. Nach Überschreiten dieser Zeit erfolgt ein Abbruch des Jobs (Abbruch-Code 322).

TIME=NOLIMIT oder (veraltet) **TIME=1440**

Gibt an, dass dieser Job ohne CPU-Zeitlimit ablaufen darf. Außerdem ist das Wait-State-Limit außer Kraft gesetzt. Diese Angabe sollte nie für ‘normale‘ Jobs verwendet werden.

Standardannahme

Der Systemprogrammierer legt fest, welcher Standardwert (Default) bei Weglassen dieses Parameters gilt.

Hinweise

Wie viel CPU-Zeit in einem Job benötigt wird, ist nur bei Kenntnis der in diesem Job benutzten Programme und der zu verarbeitenden Datenmengen abschätzbar. Bei produktiven Jobs hat man in der Regel Erfahrungswerte aus vorangegangenen Jobläufen. Bei Test-Jobs dient die Angabe des TIME-Parameters in der Regel dazu, Jobs abzubrechen, die auf Grund eines Programmierfehlers in eine Endlos-Schleife geraten sind.

Beziehungen zu anderen Parametern

TIME-Parameter in der EXEC-Anweisung

2.1.9 Schlüsselwortparameter: REGION

Verwendung

Der REGION-Parameter ermöglicht die Angabe des Speicherbedarfs bzw. Speicherlimits für alle Steps dieses Jobs.

Parametertyp

Schlüsselwortparameter, optional

REGION=nnnnnnnK oder REGION=mmmmM

Gibt an, wie viele Kilobyte (0…2096128) bzw. Megabyte (0…2047) Speicher dieser Job maximal benutzen darf.

Wirkung

Diese Angabe beeinflusst die Speicherzuordnung sowohl im unteren Bereich (unterhalb der 16 MB-Linie) als auch im oberen Bereich (oberhalb der 16 MB-Linie). Bei nachfolgenden Angaben wird zur Verfügung gestellt:

0M	gesamter Speicher im virtuellen Adressraum
<=16M	Speicher in der angegebenen Menge im unteren Bereich. Zusätzlich standardmäßig Speicher in der Größe von 32MB im oberen Bereich
>16M und <= 32M	aller verfügbarer Speicher im unteren Bereich. Zusätzlich standardmäßig Speicher in der Größe von 32MB im oberen Bereich
>32M	aller verfügbarer Speicher im unteren Bereich. Zusätzlich Speicher in der angegebenen Menge im oberen Bereich

Standardannahme

Der Systemprogrammierer legt fest, welcher Standardwert (Default) bei Weglassen dieses Parameters gilt.

Hinweise

Wie viel Speicher in einem Job benötigt wird, ist nur bei Kenntnis der in diesem Job benutzten Programme und der zu verarbeitenden Datenmengen abschätzbar. Auf die Angabe dieses Parameters kann in vielen Fällen verzichtet werden.

Beziehungen zu anderen Parametern

REGION-Parameter in der EXEC-Anweisung

2.1.10 Schlüsselwortparameter: COND

Verwendung

Der COND-Parameter ermöglicht die Angabe einer oder mehrerer Bedingungen, bei deren Auftreten der Joblauf vorzeitig beendet werden soll.

Parametertyp

Schlüsselwortparameter, optional

COND(code,op) oder COND((code,op)[,(code,op)]...)

Gibt die Bedingung(en) zur vorzeitigen Beendigung des Joblaufes an.

Standardannahme

Bei Weglassen dieses Parameters werden alle Steps im Job der Reihe nach ausgeführt. Wenn jedoch in einem Step ein Abbruch auftritt, werden standardmäßig alle nachfolgenden Steps unterdrückt.

Beziehungen zu anderen Parametern

COND-Parameter in der EXEC-Anweisung

Verweis

Die COND-Parameter in der JOB- und in der EXEC-Anweisung werden gemeinsam in einem späteren Abschnitt behandelt.

2.1.11 Schlüsselwortparameter: NOTIFY

Verwendung

Der NOTIFY-Parameter ermöglicht die Angabe einer TSO-Benutzerkennung [TSO-Userid]. Der hier angegebene Benutzer erhält eine Nachricht bei Jobende.

Parametertyp

Schlüsselwortparameter, optional

NOTIFY=userid

Gültige TSO-Benutzerkennungen sind maximal 7 Stellen lang.
Bei der Angabe &SYSUID wird der Inhalt der Systemvariablen &SYSUID, welche die aktuelle user-id beinhaltet, eingesetzt.

Standardannahme

Bei Weglassen dieses Parameters erfolgt bei Jobende keine Benachrichtigung.

2.1.12 Schlüsselwortparameter: BYTES, LINES, PAGES

Verwendung

Die Parameter BYTES, LINES, PAGES ermöglichen die Angabe eines Druckausgabe-Limits dieses Jobs. Bei Überschreiten eines der Limits soll der Joblauf vorzeitig beendet werden.

Parametertyp

Schlüsselwortparameter, optional

BYTES=zahl

Gibt an, wie viele Bytes an Druckausgabe multipliziert mit 1000 dieser Job maximal benutzen darf. Nach Überschreiten dieser Menge erfolgt ein Abbruch des Jobs (Abbruch-Code 722).

LINES=zahl

Gibt an, wie viele Zellen an Druckausgabe multipliziert mit 1000 dieser Job maximal benutzen darf. Nach Überschreiten dieser Menge erfolgt ein Abbruch des Jobs (Abbruch-Code 722).

PAGES=zahl

Gibt an, wie viele Seiten an Druckausgabe dieser Job maximal benutzen darf. Nach Überschreiten dieser Menge erfolgt ein Abbruch des Jobs (Abbruch-Code 722).

2.2 EXEC-Anweisung

2.2.1 Überblick

Verwendung

Die EXEC-Anweisung definiert den Beginn eines Steps. In einer EXEC-Anweisung wird immer ein Programm oder eine JCL-Prozedur aufgerufen und deren Verarbeitungsmerkmale definiert. Pro Step muss eine EXEC-Anweisung vorhanden sein. Eine EXEC-Anweisung innerhalb eines Jobs bedeutet das Ende des vorherigen Steps. Es sind maximal 255 Steps pro Job erlaubt.

Namensfeld

Das Namensfeld enthält den Stepnamen. Der Stepname ist optional und prinzipiell frei wählbar. Es ist jedoch sinnvoll, jedem Step einen innerhalb des Jobs eindeutigen und möglichst sprechenden Namen zu geben. Oft wird der Stepname benötigt, um Rückbezugnahmen zu formulieren.
Spalte 1 und 2 enthalten ‘//‘. Der Stepname muss in Spalte 3 beginnen. Er darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A. ..Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0... 9) oder eines der Sonderzeichen § # $ sein. Dem Stepnamen muss mindestens eine Leerstelle folgen.

Operationsfeld

Der Operationscode muss EXEC lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt. Dem Operationscode muss mindestens eine Leerstelle folgen.

Parameterfeld

Das Parameterfeld enthält einen Positions- und mehrere Schlüsselwortparameter. Ein Positionsparameter zum Aufruf eines Programms (PGM) oder einer JCL-Prozedur (PROC) muss vorhanden sein. Die Schlüsselwortparameter folgen dem Positionsparameter und sind wahlweise.

2.2.2 Positionsparameter: PGM

Verwendung

Der PGM-Parameter ermöglicht den Aufruf eines beliebigen auszuführenden Programms. Das Programm muss in ausführbarer Form, d.h. als Lademodul zur Verfügung stehen. Das Laden und Ausführen des Programms wird von der Betriebssystem-Komponente Initiator vorgenommen.

Parametertyp

Positionsparameter, hat jedoch das Aussehen eines Schlüsselwortparameters. Die Angabe eines der Parameter PGM oder PROC ist erforderlich.

PGM=progname

Hier ist der Name des auszuführenden Programmes angegeben. Er darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A. ..Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0.. .9) oder eines der Sonderzeichen § # $ sein. Das Programm muss sich als Member in einer Programmbibliothek für Lademodule befinden.

Verweise

Das Verfahren zum Erstellen eines Programms in ausführbarer Form, d.h. als Lademodul, wird in einem späteren Abschnitt kurz angerissen. Das Verfahren zum Auffinden eines ausführbaren Programms in einem späteren Abschnitt behandelt. Vorläufig wird davon ausgegangen, dass das ausführbare Programm ‘irgendwie‘ gefunden wird.

2.2.3 Positionsparameter: PROC

Verwendung

Der PROC-Parameter ermöglicht den Aufruf einer beliebigen JCL-Prozedur.

Parametertyp

Positionsparameter, kann jedoch das Aussehen eines Schlüsselwortparameters haben. Die Angabe eines der Parameter PGM oder PROC ist erforderlich.

PROC=procname oder (nur) procname

Hier ist der Name der auszuführenden JCL-Prozedur angegeben. Er darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A. ..Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A…Z, 0…9) oder eines der Sonderzeichen § # $ sein. Die Prozedur muss eine Instream-Prozedur sein oder katalogisiert in einer Prozedur-Bibliothek zur Verfügung stehen.

Verweise

Das Verfahren zum Erstellen einer JCL-Prozedur wird in einem späteren Abschnitt behandelt. Auch das Verfahren zum Auffinden einer JCL-Prozedur wird in einem späteren Abschnitt angesprochen. Vorläufig werden Prozeduren nicht benutzt.

2.2.4 Schlüsselwortparameter: PARM

Verwendung

Mit dem PARM-Parameter können beliebige Informationen bzw. Werte an das auszuführende Programm übergeben werden. Dazu muss das Programm jedoch für die Verarbeitung dieser Informationen entsprechend programmiert sein.

Parametertyp

Schlüsselwortparameter, optional

PARM=parmstring

Der an das Programm zu übergebende ‘parmstring‘ darf max. 100 Zeichen lang sein. Werden mehrere Subparameter als ‘parmstring‘ verwendet, so müssen diese durch ein Komma voneinander getrennt und durch Klammerung oder durch Hochkommata zu einem Gesamt-Parameter zusammengefasst werden.

Sonderzeichen

Bei der Verwendung von Sonderzeichen ist die entsprechende Zeichenfolge (gesamter Parameter oder der jeweilige Subparameter) in Hochkommata zu setzen. Die Sonderzeichen ‘ und & müssen grundsätzlich verdoppelt werden.

Hinweise

Welche ‘parmstring‘-Angaben in einem Step zulässig sind oder benötigt werden, ist nur bei Kenntnis des in diesem Step benutzten Programms möglich.

2.2.5 Schlüsselwortparameter: TIME

Verwendung

Der TIME-Parameter ermöglicht die Angabe eines Zeitlimits für den CPU-Zeitbedarf dieses Steps. Ist jedoch der TIME-Parameter in der JOB-Anweisung angegeben, wird dieser zuerst ausgewertet. Ist in der JOB-Anweisung TIME=NOLIMIT oder (veraltet) TIME= 1440 angegeben, so wird der TIME-Parameter in der EXEC-Anweisung ignoriert.
Ist in der JOB-Anweisung eine andere TIME-Angabe gemacht, so wird zuerst auf ein Überschreiten des Zeitlimits der JOB-Anweisung und dann auf ein Überschreiten des Zeitlimits der EXEC-Anweisung geprüft.

Parametertyp

Schlüsselwortparameter, optional

TIME=([minuten] [,sekunden])

Gibt an, wie viele Minuten (1…357912) und/oder Sekunden (1…59) an CPU-Zeit dieser Step maximal benutzen darf. Nach Überschreiten dieser Zeit erfolgt ein Abbruch des Steps (Abbruch-Code 322).

TIME=MAXIMUM

Gibt an, dass dieser Step 357912 Minuten an CPU-Zeit maximal benutzendarf. Nach Überschreiten dieser Zeit erfolgt ein Abbruch des Steps (Abbruch-Code 322).

TIME=NOLIMIT oder (veraltet) TIME=1440

Gibt an, dass dieser Step ohne CPU-Zeitlimit ablaufen darf. Außerdem ist das Wait-State-Limit außer Kraft gesetzt. Diese Angabe sollte nie für ‘normale‘ Steps verwendet werden.

Standardannahme

Der Systemprogrammierer legt fest, welcher Standardwert (Default) bei Weglassen dieses Parameters gilt.

Hinweise

Wie viel CPU-Zeit in einem Step benötigt wird, ist nur bei Kenntnis des in diesem Job benutzten Programms und der zu verarbeitenden Datenmengen abschätzbar.

Beziehungen zu anderen Parametern

TIME-Parameter in der JOB-Anweisung

2.2.6 Schlüsselwortparameter: REGION

Verwendung

Der REGION-Parameter ermöglicht die Angabe des Speicherbedarfs bzw. Speicherlimits für diesen Step. Ist jedoch der REGION-Parameter in der JOB-Anweisung angegeben, so hat dieser Vorrang, d.h. der REGION-Parameter in der EXEC-Anweisung wird ignoriert.

Parametertyp

Schlüsselwortparameter, optional

REGION=nnnnnnnK oder REGION=mmmmM

Gibt an, wie viele Kilobyte (0…2096128) bzw. Megabyte (0…2047) Speicher dieser Step maximal benutzen darf.

Wirkung

Diese Angabe beeinflusst die Speicherzuordnung sowohl im unteren Bereich (unterhalb der 16MB-Linie) als auch im oberen Bereich (oberhalb der 16MB-Linie). Die genaue Wirkung dieses Parameters ist die gleiche, wie beim REGION-Parameter der JOB-Anweisung.

Standardannahme

Der Systemprogrammierer legt fest, welcher Standardwert (Default) bei Weglassen dieses Parameters gilt.

Hinweise

Wie viel Speicher in einem Step benötigt wird, ist nur bei Kenntnis des in diesem Step benutzten Programms und der zu verarbeitenden Datenmengen abschätzbar. Auf die Angabe dieses Parameters kann in vielen Fällen verzichtet werden.

Beziehungen zu anderen Parametern

REGION-Parameter in der JOB-Anweisung

2.2.7 Schlüsselwortparameter: COND

Verwendung

Der COND-Parameter ermöglicht die Angabe einer oder mehrerer Bedingungen, bei deren Auftreten der aktuelle Step unterdrückt oder ausgeführt werden soll.

Parametertyp

Schlüsselwortparameter, optional

COND=(code,op[,stepname])

COND=EVEN

COND=ONLY

COND=((code,op[,stepname])[,(code,op[,stepname])]...[,EVEN])

COND=((code,op[,stepname])[,(code,op[,stepname])]...[,ONLY])

Gibt die Bedingung(en) zum Unterdrücken oder Ausführen des aktuellen Steps an. Ist jedoch der COND-Parameter in der JOB-Anweisung angegeben, wird dieser zuerst ausgewertet und hat somit Vorrang vor dem COND-Parameter in der EXEC-Anweisung.

Standardannahme

Bei Weglassen dieses Parameters wird der aktuelle Step ausgeführt. Wenn jedoch in einem vorangegangenen Step ein Abbruch auftrat, wird standardmäßig der aktuelle Step unterdrückt.

Beziehungen zu anderen Parametern

COND-Parameter in der JOB-Anweisung

Verweis

Die COND-Parameter in der JOB- und in der EXEC-Anweisung werden gemeinsam in einem späteren Abschnitt behandelt.

3. Datenbestandsbeschreibung (1. Teil)

3.1 DD-Anweisung (einfache Formen)

3.1.1 Überblick

Verwendung

Bei der Ausführung eines Steps bzw. eines Programms werden in der Regel Datenbestände benötigt. Eine DD-Anweisung [Data Definition] beschreibt einen Datenbestand.
Datenbestände können nach verschiedenen Kriterien gruppiert werden:

Art der Verwendung

• Eingabe-Datenbestände	Input Files	nur Lesen
• Ausgabe-Datenbestände	Output Files	nur Schreiben
• Änderungs-Datenbestände	Update Files	Lesen und Zurück-Schreiben
• Arbeits-Datenbestände	Work Files	zuerst Schreiben, später Lesen

Art des Datenträgers

- Druckerliste
- Magnetband
- Magnetplatte
- ‘moderne‘ magnetische/optische Datenträger, die in der Regel ähnlich wie Magnetplatten gehandhabt werden

Lebensdauer des Datenbestandes

- permanenter Datenbestand, d.h. er existiert vor Jobbeginn und / oder nach Jobende
- temporärer Datenbestand, d.h. er existiert nur innerhalb *eines* Jobs
- Systemeingabe, d.h. der Datenbestand wird vor Jobbeginn von JES2/3 verwaltet und wird während des Jobs (im wörtlichen Sinne) verarbeitet
- Systemausgabe, d.h. der Datenbestand wird während des Jobs erzeugt und wird nach Jobende von JES2/3 in einer Output Queue verwaltet. Letztlich wird dieser Datenbestand gedruckt, an ein anderes Rechnersystem weitergeleitet oder gelöscht.

3.1.2 Bezugnahme zum Programm

Zweck

Die DD-Anweisung stellt die logische Verknüpfung zwischen einem physischen Datenbestand und dem Programm her. Außerdem beschreibt sie physische und logische Eigenschaften eines Datenbestandes.
Bei JES2 ordnet der Initiator die jeweiligen Datenbestände stepweise zu. Bei JES3 werden sie von JES3 selbst in der Main Device Scheduling-Phase zugeordnet.

Allgemeine Regel

Pro benötigtem Datenbestand muss eine DD-Anweisung vorhanden sein.
Die DD-Anweisungen eines Steps können in der Regel in beliebiger Reihenfolge nach der EXEC-Anweisung des Jobsteps, dem sie zugeordnet werden sollen, angegeben werden.

Beschreibung eines Datenbestandes

In einer DD-Anweisung wird u.a. folgendes angegeben:

- Name des Datenbestandes
- Handelt es sich um einen Systemeingabe-Datenbestand oder Systemausgabe-Datenbestand?
- Wird der Datenbestand neu erstellt oder existiert er bereits?
- Was soll mit dem Datenbestand nach der Ausführung geschehen. Soll er aufbewahrt oder gelöscht werden?
- Wie ist der Typ und der Name des Datenträgers?
- Wie viel Speicherplatz benötigt er?
- usw.

Verknüpfung zum Programm

- ASSEMBLER — der Name in der DD-Anweisung ergibt sich aus dem DDNAME-Parameter im DCB-Makro
- COBOL — der Name in der DD-Anweisung ergibt sich aus der Angabe nach ‘ASSIGN TO‘
- PL/1 — der Name in der DD-Anweisung ergibt sich aus der Angabe bei TITLE (wenn vorhanden) bzw. bei FILE

3.1.3 Syntax

Namensfeld

Das Namensfeld enthält den DD-Namen. Der DD-Name ist in der Regel vom benutzten Programm fest vorgegeben und muss so angegeben werden, wie er im Programm definiert wurde.
Spalte 1 und 2 enthalten '//'. Der DD-Name muss in Spalte 3 beginnen. Er darf maximal 8 Zeichen (Stellen) lang sein. Das erste Zeichen muss alphabetisch (A...Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0.. .9) oder eines der Sonderzeichen § # $ sein. Dem DD-Namen muss mindestens ein Blank folgen.

Operationsfeld

Der Operationscode muss DD lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt. Dem Operationscode muss mindestens ein Blank folgen.

Parameterfeld

Das Parameterfeld enthält Positions- und Schlüsselwortparameter. Benötigte Schlüsselwortparameter folgen dem Positionsparameter (falls vorhanden). Welche Parameter tatsächlich zu benutzen sind, hängt von der Art des Datenbestandes ab.

3.2 Systemeingabe-Datenbestände (Instream-Datenbestände)

3.2.1 Positionsparameter * bzw. DATA

Verwendung

Systemeingabe-Datenbestände (Instream-Datenbestände) sind Datenbestände, die vor Jobbeginn von JES2/3 verwaltet und während des Jobs verarbeitet werden. Sie stehen danach nicht länger zur Verfügung.
Die zu einem Datenbestand dieser Art gehörenden Daten wurden früher in Form von Lochkarten (‘Datenkarten‘) zwischen die JCL-Anweisungen (‘JCL-Karten‘) eingebettet. Daher stammt auch die Bezeichnung ‘Instream-Daten‘, also Daten, die im Strom der Lochkarten eingebettet sind.
Heute hat man keine Lochkarten mehr. Das Konzept der zwischen die JCL-Anweisungen eingebetteten 80-stelligen Datensätze existiert jedoch noch immer.
Ein Instream-Datenbestand wird vom JES2/3 beim Einlesen (Input-Phase) in einer Spool-Datei zwischengespeichert.

Abgrenzung JCL-Karten – Datenkarten

Die Datensätze müssen von den JCL-Karten abgegrenzt werden.
Der Beginn der Datensätze wird durch die DD-Anweisung selbst gekennzeichnet. Dazu dienen die Positionsparameter ‘*‘ oder (alternativ) ‘DATA‘. Der Datensatz unmittelbar nach der DD-Anweisung ist der erste Datensatz.
Das Ende der Datensätze wird durch eine spezielle JCL-Anweisung, eine Delimiter-Anweisung (in der Regel /* in Spalten 1 bis 2), gekennzeichnet. Der Datensatz unmittelbar vor der Delimiter-Anweisung ist der letzte Datensatz.
In bestimmten Situationen kann die spezielle Delimiter-Anweisung entfallen. Das Ende der Datenkarten wird dann durch das Auftreten einer normalen JCL-Anweisung (‘//‘ in Spalten 1 bis 2) gekennzeichnet.

Parametertyp

Positionsparameter. Die Angabe eines der Parameter * oder DATA ist zur Definition eines Instream-Datenbestandes erforderlich.

* bzw. DATA

Hiermit wird der Beginn eines Instream-Datenbestandes gekennzeichnet.

Verweise

Weitere mögliche Parameter (Schlüsselwortparameter) bei Instream-Datenbeständen sind DLM=, DCB=, LRECL=.

3.2.2 Schlüsselwortparameter: DLM

Begrenzung bei Systemeingabe-Datenbeständen

Es ist eine Methode zur Begrenzung (Kennzeichnung des Endes) eines Systemeingabe-Datenbestandes (Instream-Datenbestandes) erforderlich.
Im einfachsten Fall wird das Ende des Instream-Datenbestandes implizit durch das Auftreten einer normalen JCL-Anweisung (’//’ in Spalten 1 bis 2) gekennzeichnet.
Das Ende des Instream-Datenbestandes kann auch explizit durch das Auftreten einer speziellen JCL-Anweisung, einer Delimiter-Anweisung (in der Regel ’/*’ in Spalten 1 bis 2), gekennzeichnet werden.
Falls der Instream-Datenbestand Datensätze beinhaltet, die wie eine JCL-Anweisung (‘//‘ in Spalten 1 bis 2) aussehen, so ist die Form
DD DATA
zu benutzen. Das Ende dieses Datenbestandes ist dann erst durch das Auftreten einer Delimiter-Anweisung (in der Regel ’/*’ in Spalten 1 bis 2) gegeben.
Falls der Instream-Datenbestand Datensätze beinhaltet, die wie eine JCL-Anweisung (’//’ in Spalten 1 bis 2) und/oder wie die spezielle Delimiter-Anweisung (/* in Spalten 1 bis 2) aussehen, so ist die Form
DD DATA,DLM=xx
zu benutzen. Das Ende dieses Datenbestandes ist dann erst durch das Auftreten einer entsprechenden Delimiter-Anweisung (mit ’xx‘ in Spalten 1 bis 2) gegeben.

Parametertyp

Schlüsselwortparameter, optional

DLM=xx

Als Delimiter ist eine beliebige Folge von genau zwei Zeichen zu definieren, die in den Instream-Daten in den Spalten 1 bis 2 nicht vorkommt.

Sonderzeichen

Bei der Verwendung von Sonderzeichen ist die Delimiter-Zeichenfolge Hochkommata zu setzen. Die Sonderzeichen ’ und & müssen grundsätzlich verdoppelt werden.

Übersicht: Erkennen des Endes der Instream-Daten

DD-Anweisung	JES2	JES3	Bemerkung
DD *	/* oder //	*/ oder //	Normalfall
DD *,DLM=xx	xx oder //	xx	vermeiden!!
DD DATA	/*	/*	
DD DATA,DLM=xx	xx	xx	

3.3 Systemausgabe-Datenbestände (SYSOUT, Druckausgabe)

3.3.1 Schlüsselwortparameter: SYSOUT, HOLD

Verwendung

Systemausgabe-Datenbestände (SYSOUT) sind Datenbestände, die bei Jobbeginn von JES2/3 angelegt und in die während des Jobs Daten geschrieben werden können.
Die Daten stehen danach zum Ausdrucken (Druckausgabe-Datenbestände), zum Anschauen, zum Archivieren oder auch zum Löschen zur Verfügung.
Die zu einem Datenbestand dieser Art gehörenden Datensätze wurden früher in der Regel mit Zeichen orientierten Druckern ausgegeben. Dabei wird jeder Datensatz als eine Druckzeile ausgegeben. An Steuerungsmöglichkeiten gibt es nur den Seitenvorschub (Positionieren des Endlospapiers auf den Anfang einer neuen Seite), den ein- oder mehrzeiligen Zeilenvorschub und die Vorschubunterdrückung.
Die Datensätze eines Druckausgabe-Datenbestandes bestehen deshalb

- aus einem Byte (Byte 1), das verschlüsselte Steuerinformationen für den Drucker beinhaltet,
- aus einer Folge von Bytes (ab Byte 2), welche die tatsächlich zu druckenden Daten des Satzes beinhaltet.

Da Drucker dieser Art meist 132 Druckstellen pro Zeile aufweisen, haben solche Datenbestände oft eine Satzlänge von 133 Bytes (LRECL=133).
Ein SYSOUT-Datenbestand wird vom JES2/3 in einer Spool-Datei zwischengespeichert.

Ausgabeklassen

Der Datenbestand wird gemäß einer Ausgabeklasse in eine Ausgabe-Warteschlange [Job Output Queue] zur Weiterverarbeitung eingeordnet.
Eine Ausgabeklasse symbolisiert die Verarbeitungsmerkmale, z.B. die Zuordnung zu einer bestimmten Ausgabeeinheit (z.B. Drucker) oder die Art des zu verwendenden Papiers (Einfach-, Mehrfach-Papier).

HOLD-Status

In einer Installation können Ausgabeklassen (und damit Ausgabe-Warteschlangen) als gehaltene Klassen‘ (HOLD-Klassen) definiert werden. Druckausgabe-Datenbestände in diesen Warteschlangen sind nicht für das tatsächliche Drucken freigegeben.
Alternativ können auch einzelne Druckausgabe-Datenbestände in den HOLD-Status gesetzt werden.
Somit kann der Datenbestand z.B. auf Richtigkeit und Vollständigkeit geprüft werden, bevor er zum Drucken freigegeben oder gelöscht wird.

Parametertyp

Schlüsselwortparameter, optional. Der SYSOUT-Parameter ist zur Definition eines SYSOUT-Datenbestandes erforderlich. Der HOLD-Parameter kann zusätzlich benutzt werden.

SYSOUT=([class][,programm][,formname])

Hiermit werden Eigenschaften des SYSOUT-Datenbestandes definiert.

class

Gültige Ausgabeklassen sind A-Z und 0-9. Die Eigenschaften einer bestimmten Klasse müssen in jeder Installation dokumentiert sein.
Die Angabe von '*' bedeutet, dass dieselbe Ausgabeklasse verwendet werden soll, die im MSGCLASS-Parameter der JOB-Anweisung spezifiziert ist.

programm

Nur in Sonderfällen verwendet. Hier kann ein spezielles Programm zur Steuerung der Ausgabe spezifiziert werden.
INTRDR ist ein spezielles Programm zum Umleiten der Ausgabe in den Internal Reader als Job-Eingabe.

formname

Optional. Gibt den Namen eines Formulars an, das zum Drucken benutzt werden soll. Die Namen und Eigenschaften der Formulare müssen in jeder Installation dokumentiert sein.

HOLD=[Y | YES | N | NO]

Der Druckausgabe-Datenbestand soll bzw. soll nicht in den HOLD-Status gesetzt werden. Default ist NO.

Verweise

Weitere mögliche Parameter (Schlüsselwortparameter) bei SYSOUT-Datenbeständen sind COPIES, DCB, RECFM, LRECL, OUTPUT. Weiterhin bestehen Wechselbeziehungen zur OUTPUT-Anweisung und deren Parameter.

3.3.2 Schlüsselwortparameter: COPIES

Erstellen von Kopien

Es ist möglich, dem JES2/3 den Auftrag zu erteilen, den Druckvorgang mehr oder nach einer bestimmten Systematik auszuführen.
Bei Zeichen- bzw. Zeilen orientierten Druckern ist es nur möglich, den Druckvorgang mehrfach ausführen zu lassen, d.h. mehrere Kopien eines Datenbestandes zu drucken.
Bei modernen Seiten orientierten Druckern (z.B. Laser-Druckern) wird in der Regel auf einheitlich weißem Papier gedruckt. Allerdings können aufgrund der verwendeten Drucktechnik keine Durchschreibe-Formularsätze benutzt werden. Dafür können mit Hilfe des COPIES-Parameters mehrere Gruppen von Kopien erzeugt werden.

Parametertyp

Schlüsselwortparameter, optional.

COPIES=nnn

Hiermit wird die Anzahl der Kopien bei zeichen- bzw. zeilenorientierten Druckern definiert (nnn ist 1 . . . 254). Der Druckvorgang wird einmal komplett durchgeführt, bevor er entsprechend dieser Anzahl wiederholt wird.

COPIES=([nnn],(group-value[,group-value],...))

Hiermit wird die Anzahl der Gruppen und die Anzahl der Kopien pro Gruppe bei Seiten orientierten Druckern (z.B. Laser-Druckern) definiert (nnn kann zwar angegeben werden, wird aber ignoriert). Der Druckvorgang wird so oft ausgeführt, wie Gruppen (max. 8) angegeben sind. Innerhalb einer Gruppe wird durch ‘group-value angegeben, wie oft jede Seite unmittelbar nacheinander gedruckt werden soll. Die Summe der Gruppenwerte darf max. 254 sein.

Verweise

Diese Angaben können auch mit Hilfe der OUTPUT-Anweisung definiert werden.

3.4 Unterdrückung der Ein-/Ausgabe (Scheindateien)

3.4.1 Positionsparameter: DUMMY

Verwendung

Scheindateien sind Datenbestände, die physisch nicht existieren, die jedoch zum problemlosen Ablauf eines Programms notwendig sind. Würde man entsprechende DD-Anweisungen einfach weglassen, wäre in der Regel ein Programmabsturz die Folge.
Bei einer Definition als Scheindatei (DUMMY-Datenbestand) werden physische Ein-/Ausgabeoperationen nicht durchgeführt. Für Ausgabedateien ergibt sich der Effekt, dass das Programm zwar problemlos WRITE-Operationen durchführt, die auszugebenden Daten jedoch im Nichts (im Papierkorb) landen. Als mögliche Anwendung bietet sich die Unterdrückung von unerwünschten Ausgabedaten an.
Bei Eingabedateien wird beim ersten Versuch des Programmes, eine READ-Operation durchzuführen, die Dateiende-Bedingung [End-of-File] ausgelöst. Üblicherweise reagieren die Programme darauf, indem sie die Datei-Verarbeitung beenden. Eine mögliche Anwendung bietet sich bei Programmtests, wenn die grundsätzliche Funktionsfähigkeit eines Programms geprüft werden soll, die zu verarbeitenden Daten aber (noch) nicht vorliegen.

Parametertyp

Positionsparameter. Die Angabe ist zur Definition einer Scheindatei erforderlich.

DUMMY

Hiermit wird eine Datei als Scheindatei gekennzeichnet.

Verweise

Weitere mögliche Parameter (Schlüsselwortparameter) bei Scheindateien sind DCB=, LRECL=.

3.5 Einfache Anwendungen

3.5.1 Datenbestand kopieren mit IEBGENER (einfache Formen)

Verwendung

Eine häufig durchzuführende Anwendung besteht im Kopieren eines Datenbestandes in einen zweiten Datenbestand. Für diese Standard-Anwendung steht das Dienstprogramm IEBGENER zur Verfügung. In der einfachsten Form kann IEBGENER einen sequentiellen Datenbestand in einen anderen sequentiellen Datenbestand kopieren.

Programmaufruf

Programmname ist IEBGENER. Der Programmaufruf lautet:

```
//stepname EXEC PGM=IEBGENER
```

Datei-Beschreibungen

Das Programm IEBGENER benötigt 4 DD-Anweisungen.

SYSUT1

Beschreibung der Eingabedatei. Beliebiger sequentieller Datenbestand (und somit Definition als Instream-Datenbestand möglich), z.B.:

```
//SYSUT1 DD *
Datensatz 1
Datensatz 2
Datensatz 3
/*
```

SYSUT2

Beschreibung der Ausgabedatei. Beliebiger sequentieller Datenbestand (und somit Definition als Druckausgabe-Datenbestand möglich), z.B.:

```
//SYSUT2   DD SYSOUT=*
```

SYSPRINT

Beschreibung einer Protokolldatei für Nachrichten und Fehlermeldungen, als Druckausgabe gestaltet, z.B.:

```
//SYSPRINT DD SYSOUT=*
```

SYSIN

Beschreibung einer Datei für Dienstprogramm-Steueranweisungen, im Lochkartenformat gestaltet (und somit definierbar als Instream-Datenbestand). Wenn nur eine einfache 1:1-Kopieroperation durchgeführt werden soll, kann auf Dienstprogramm-Steueranweisungen verzichtet werden (und somit eine Definition als Scheindatei erfolgen), z.B.:

```
//SYSIN   DD   DUMMY
```

3.5.2 Datenbestand sortieren mit DFSORT (einfache Formen)

Verwendung

Eine häufig durchzuführende Anwendung besteht im Sortieren der Datensätze eines Datenbestandes und Speichern der sortierten Sätze in einen zweiten Datenbestand. Für diese Standard-Anwendung steht das Programmprodukt DFSORT zur Verfügung. In der einfachsten Form kann DFSORT die Datensätze eines sequentiellen Datenbestandes sortieren und in einen anderen sequentiellen Datenbestand speichern.

Programmaufruf

Programmname ist ICEMAN (alternativ SORT) Der Programmaufruf lautet:

```
//stepname EXEC PGM=ICEMAN oder PGM=SORT
```

Datei-Beschreibungen

Das Programm ICEMAN benötigt 4 DD-Anweisungen.

SORTIN

Beschreibung der Eingabedatei. Beliebiger sequentieller Datenbestand (und somit Definition als Instream-Datenbestand möglich), z.B.:

```
//SORTIN   DD   *
Datensatz X
Datensatz A
Datensatz M
/ *
```

SORTOUT

Beschreibung der Ausgabedatei. Beliebiger sequentieller Datenbestand (und somit Definition als Druckausgabe-Datenbestand möglich), z.B.:

```
//SORTOUT  DD SYSOUT=*
```

SYSOUT

Beschreibung einer Protokolldatei für Nachrichten und Fehlermeldungen, als Druckausgabe gestaltet, z.B.:

```
//SYSOUT   DD SYSOUT=*
```

SYSIN

Beschreibung einer Datei für Steueranweisungen, im Lochkartenformat strukturiert (und somit definierbar als Instream-Datenbestand). Hiermit werden die Einzelheiten für den Sortiervorgang beschrieben, z.B.:

```
//SYSIN   DD *
SORT FIELDS=(1,11,CH,A)
/*
```

3.5.3 Steueranweisung: SORT

Sortieren von Datensätzen

Zum Sortieren von Datensätzen sind ein oder mehrere Sortierfelder zu definieren. In *einem* Sortierfeld sind die Daten enthalten, die für die aufsteigende oder absteigende Sortierfolge entscheidend sind.
Es ist möglich, mehrere Sortierfelder zu definieren, die allerdings in einer Hierarchie zueinander stehen müssen.

Hauptsortierfeld, 1. Untersortierfeld, 2. Untersortierfeld, usw.

Für jedes Sortierfeld, zumindest für ein Sortierfeld, sind folgende Angaben nötig:

- Anfangsposition des Sortierfeldes (Byteposition im Datensatz)
- Länge des Sortierfeldes (in Bytes)
- Datenformat im Sortierfeld. Häufig CH für Zeichenformat [Character]
- gewünschte Sortierfolge. A für aufsteigend [ascending] bzw. D für absteigend [descending]

Syntax der Steueranweisung

SORT FIELDS= (position, länge, format, folge)

oder

SORT FIELDS=(position,länge, folge), FORMAT=format

4. Datenbestandsbeschreibung (2. Teil)

4.1 DD-Anweisung (weitere Formen)

4.1.1 Überblick

Verwendung

Neben den bisher behandelten Formen der DD-Anweisung zur Definition von

- Systemeingabe-Datenbeständen (Instream-Datenbeständen)
- Systemausgabe-Datenbeständen (SYSOUT, Druckausgabe-Datenbeständen)
- Scheindateien (DUMMY)

gibt es weitere Formen zur Definition von

- permanenten Datenbeständen auf Platte oder Band
- temporären Datenbeständen auf Platte oder Band.

Abhängig von der Art der Datei (Platte oder Band, bereits existent oder neu anzulegen) und ihrer Organisationsform (sequentiell, untergliedert, VSAM) ist eine unterschiedliche Anzahl von Schlüsselwortparametern in der DD-Anweisung anzugeben.

4.2 Datenbestände auf Platte oder Band

4.2.1 Schlüsselwortparameter: DSN bzw. DSNAME

Verwendung

Datenbestände auf Platte oder Band sind durch einen Datenbestandsnamen [Dataset Name] (DSN bzw. DSNAME) gekennzeichnet.
Lediglich bei temporären Datenbeständen (d.h. bei Datenbeständen, die nur innerhalb eines Jobs existieren) kann die Angabe des DSN entfallen.

Parametertyp

Schlüsselwortparameter, optional.

DSN=datasetname oder DSNAME=datasetname

Hiermit wird der Name eines sequentiellen, untergliederten oder VSAM-Datenbestands definiert.

Einstufiger Name

Ein einstufiger oder einfacher Datenbestandsname darf maximal 8 Zeichen (Stellen) lang sein. Beispiel: DSN=FILE
Das erste Zeichen muss alphabetisch (A. . .Z, Großbuchstaben!) oder eines der Sonderzeichen ~ # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0.. .9) oder eines der Sonderzeichen ~ # $ sein.
Einstufige Namen werden (fast) nicht benutzt.

Mehrstufiger (qualifizierender) Name

Ein mehrstufiger oder qualifizierender Datenbestandsname darf maximal 44 Zeichen (Stellen) lang sein. Er besteht aus einer Aneinanderreihung mehrerer einstufiger Namen, die voneinander durch Punkte getrennt werden.
Beispiel: DSN=PROD.MVS.DATA
In der Regel werden solche mehrstufigen Namen benutzt.

Name für temporären Datenbestand

Der Name eines temporären Datenbestandes besteht aus einem einfachen Namen, dem ‘&& vorangestellt wird. Beispiel: DSN=&&TEMP
Wird hier kein DSN angegeben, benutzt das System einen internen Namen.

Sinn mehrstufiger Datenbestandsnamen

Mehrstufige Namen werden aus folgenden Gründen verwendet:

- Es lassen sich wesentlich mehr unterschiedliche Datenbestandsnamen bilden als mit einfachen Namen.
- Man kann den einzelnen Stufen unterschiedliche Bedeutungen zuordnen, z.B. Bezeichnung einer Organisationseinheit, Kennzeichnung eines Anwendungsgebietes, Art des Inhalts, usw.
- Beispiel: DSN=FILIALE.HAMBURG.NONFOOD.UMSATZ
- In der Regel kommt der ersten (höchsten) Namensstufe eine besondere Bedeutung bei. Sie wird High-Level-Qualifier (HLQ) genannt. Dieser HLQ spielt für die system-interne Verwaltung sowie für die Definition der Zugriffsrechte auf diesen Datenbestand eine wichtige Rolle. Oft muss der HLQ identisch sein mit der TSO-Userid des Erstellers oder ‘Besitzers‘ des Datenbestandes. Beispiel, wenn der TSO-Userid des Erstellers SKIS99T ist: DSN=SKIS99T.MVS.FILE
- Durch eine Regelung dieser Art ist automatisch sichergestellt, dass verschiedene Personen keine gleichnamigen Datenbestände definieren können.
- Manchmal kommt auch der letzten (niedrigsten) Namensstufe eine besondere Bedeutung bei. sie wird Low-Level-Qualifier (LLQ) genannt. Durch diesen LLQ wird die Art des Dateiinhaltes gekennzeichnet. Insbesondere TSO und ISPF orientieren sich an dieser Konvention. Beispiel, wenn der Inhalt des Datenbestandes aus einem COBOL-Programm bestehen soll: DSN=SKIS99T.MVS.COBOL

Hinweis

Beim Arbeiten mit ISPF sind dreistufige Namen besonders praktisch zu handhaben. Namen mit mehr oder weniger Stufen sind aber auch erlaubt.

Sonderform: DSN=NULLFILE

Mit dieser Sonderform wird eine Scheindatei (ein DUMMY-Datenbestand) angegeben. Die Wirkung entspricht der einer DUMMY DD-Anweisung.

Sonderform: DSN=datasetname(membername)

Mit dieser Sonderform wird ein untergliederter Datenbestand und (innerhalb der Klammern) *ein bestimmtes Element* (ein bestimmtes Member) daraus angegeben.

Beispiele: DSN=PROD.MVS.LIB(MEM1) oder DSN=&&LIB(MEM1)

Diese Form ist dann sinnvoll, wenn ein Programm, das eigentlich zum Bearbeiten eines sequentiellen Datenbestandes dient, statt dessen ein Member eines *untergliederten* Datenbestands bearbeiten soll.

4.2.2 Schlüsselwortparameter: DISP

Verwendung

Die ‘Disposition‘ eines Datenbestandes auf Platte oder Band gibt Auskunft über seinen Zustand bzw. über durchzuführende Aktionen vor Stepbeginn und nach Stepende.
Ein eventuell durch den DISP-Parameter spezifiziertes Anlegen eines Datenbestandes geschieht immer bei Stepbeginn, ein Löschen geschieht immer bei Stepende.

Parametertyp

Schlüsselwortparameter, optional.

DISP=([status][,normalend][,abnormalend])

Hiermit wird der Status (Zustand vor Stepbeginn), die Aktion bei normalem Stepende (kein Abbruch [ABEND] im Step) und die Aktion bei abnormalem Stepende (nach Abbruch [ABEND] im Step) beschrieben.
Die tatsächliche Wirkung hängt in einigen Fällen ab von der Organisationsform des Datenbestandes (sequentiell, untergliedert, VSAM).
Das Weglassen einzelner Unterparameter ist möglich. Das System benutzt dann Standardannahmen (Default-Werte). Werden alle Unterparameter weggelassen, gilt DISP=(NEW,DELETE,DELETE).
Die Standardannahmen ergeben für den ungeübten Benutzer manchmal etwas überraschende Wirkungen. Es ist somit eindeutiger, wenn zumindest die beiden ersten Unterparameter auch tatsächlich angegeben werden.

status: NEW, OLD, SHR, MOD

Hiermit wird der Status eines Datenbestandes (Zustand vor Stepbeginn) beschrieben und angegeben, ob der Datenbestand

- exklusiv (d.h. ausschließlich für diesen Job) oder
- shared (d.h. gleichzeitig auch für andere Jobs)

benutzbar (im Zugriff) sein soll.

- OLD Zugriff exklusiv
  Zustand: Datenbestand existiert bereits bei Stepbeginn
- SHR Zugriff shared
  Zustand: Datenbestand existiert bereits bei Stepbeginn
- NEW Zugriff exklusiv
  Zustand: Datenbestand existiert nicht bei Stepbeginn, wird deshalb bei Stepbeginn vom System angelegt
- MOD Zugriff exklusiv
  Zustand: falls Datenbestand nicht bei Stepbeginn existiert, wird er bei Stepbeginn vom System angelegt. Der Datenbestand kann auch bereits bei Stepbeginn existieren.

normalend: KEEP, CATLG, UNCATLG, DELETE, PASS

Hiermit wird die Aktion bei normalem Stepende (kein Abbruch [ABEND] im Step) beschrieben. Der hier erwähnte “Katalogeintrag“ wird in einem späteren Abschnitt ausführlich behandelt.

- KEEP
  Zustand Datenbestand wird beibehalten
  nicht zulässig für temporäre Datenbestände
  Katalogeintrag bleibt unverändert
- CATLG
  Zustand: Datenbestand wird beibehalten
  nicht zulässig für temporäre Datenbestände
  Katalogeintrag wird erstellt
- UNCATLG
  Zustand: Datenbestand wird beibehalten
  nicht zulässig für temporäre Datenbestände
  Katalogeintrag wird entfernt
- DELETE
  Zustand: Datenbestand wird gelöscht
  Katalogeintrag wird nur dann gelöscht, wenn der Datenbestand mit Hilfe eines Katalogeintrages gefunden wurde.
- PASS
  Zustand: Datenbestand wird vorläufig beibehalten
  Katalogeintrag bleibt vorläufig unverändert
  erst später im Job wird die endgültige Aktion festgelegt

abnormalend: KEEP, CATLG, UNCATLG, DELETE

Hiermit wird die Aktion bei abnormalem Stepende (nach Abbruch [ABEND] im Step) beschrieben.
Es können (mit Ausnahme von PASS) die gleichen Aktionen wie bei normalem Stepende angegeben werden.
Wenn dieser Unterparameter nicht angegeben ist, wird die gleiche Aktion wie bei normalem Stepende ausgeführt.

Hinweise zur Angabe von PASS

PASS wird verwendet, wenn erst später im Job die endgültig durchzuführende Aktion festgelegt werden kann oder soll. Das ist häufig der Fall bei:

- einem Datenbestand auf Band oder Platte, wenn erst später im Job entschieden werden soll, ob er beibehalten oder gelöscht werden soll
- einem Datenbestand auf Band, der in einem Step benutzt wird und in einem späteren Step wieder benutzt werden soll. Ein zwischenzeitliches Entladen des Bandes kann damit vermieden werden
- einem temporären Datenbestand auf Band oder Platte, der über mehrere Steps hinweg beibehalten werden soll (ein KEEP oder CATLG ist unzulässig!).

Ein Datenbestand, der in einem Step mit PASS angegeben wird, kann in einem späteren Step übernommen werden. Soll er in mehreren späteren Steps übernommen werden, muss er in den zwischenliegenden Steps wieder mit PASS weitergegeben werden.

Zusatzinformationen

Bestimmte Angaben haben eine zusätzliche Wirkung.
Wirkung bei Ausgabeoperationen in sequentielle Datenbestände:

- OLD — Schreiben ab Dateianfang, d h Überschreiben eventuell vorhandener Daten
- SHR — wie OLD. Jedoch sollte SHR in der Regel nicht beim Schreiben in Datenbestände verwendet werden (Gefahr des totalen Datenverlustes)
- MOD — Schreiben ab bisherigem Dateiende, d.h. Fortschreiben (Ergänzen) eventuell vorhandener Daten

Wirkung bei Ausgabeoperationen in untergliederte Datenbestande

- OLD — Schreiben neuer Elemente und Ersetzen existierender (gleichnamiger) Elemente zulässig
- SHR — wie OLD. Jedoch sollte SHR in der Regel nicht beim Schreiben in Datenbestände verwendet werden (Gefahr des totalen Datenverlustes)
- MOD — Schreiben neuer Elemente zulässig. Der Versuch des Ersetzens existierender (gleichnamiger) Elemente führt zu einem Abbruch (ABEND)

Wirkung bei VSAM-Datenbeständen (diese müssen katalogisiert sein)

- OLD — kein gleichzeitiger Zugriff durch andere Jobs (unabhängig von den SHAREOPTIONS)
- SHR — gleichzeitiger Zugriff durch andere Jobs wird durch die SHAREOPTIONS geregelt
- MOD — nicht für VSAM-Datenbestände
- NEW — nur bei Einsatz von SMS für VSAM-Datenbestände zulässig.
  Alternative VSAM-Datenbestand mit Dienstprogramm IDCAMS anlegen
- CATLG — nur bei Einsatz von SMS für VSAM-Datenbestande zulassig.
  Alternative VSAM-Datenbestand mit Dienstprogramm IDCAMS anlegen
- UNCATLG — unzulässig bzw. unsinnig für VSAM-Datenbestande
- DELETE — nur bei Einsatz von SMS für VSAM-Datenbestände zulässig
  Alternative VSAM-Datenbestand mit Dienstprogramm IDCAMS löschen
- PASS — nur bei Einsatz von SMS für temporäre VSAM-Datenbestände zulässig

4.2.3 Schlüsselwortparameter: UNIT

Verwendung (vereinfachte Darstellung)

Die Unit‘ eines Datenbestandes spezifiziert das Ein-/Ausgabegerät (oder eine Gruppe von zulässigen Geräten) und damit auch den Datenträgertyp für das Anlegen oder das wieder Finden eines Datenbestandes.
Der UNIT-Parameter kann beim wieder Finden eines Datenbestandes entfallen, wenn ein “Katalogeintrag‘ dafür existiert. Der hier erwähnte “Katalogeintrag“ wird in einem späteren Abschnitt ausführlich behandelt.
Der UNIT-Parameter wirkt mit dem VOL-/VOLUME-Parameter zusammen.

Parametertyp

Schlüsselwortparameter, optional.

UNIT=device-type

Hiermit wird ein Gerätetyp (Datenträgertyp) für den Datenbestand ausgewählt. Es wird die Typenbezeichnung eines IBM-Gerätes angegeben. Welche Typen angegeben werden können, ist installationsabhängig und muss dokumentiert sein. Beispiel: UNIT=3390
Häufig eingesetzte Typen sind

- 3390 Magnetplattengerat
- 3380 Magnetplattengerat älterer Bauart
- 3490 Magnetband-Kassettengerät
- 3480 Magnetband-Kassettengerät älterer Bauart
- 3420 Magnetband-Rollengerat

UNIT=group-name

Hiermit wird eine Gruppe von zulässigen Geräten für den Datenbestand ausgewählt. Zu diesem Zweck werden in jeder Installation Gruppennamen (innerhalb der Installation frei wählbare Namen) definiert. Jeder davon dient als Sammelbegriff für ein oder mehrere Geräte mit gleichen oder ähnlichen Eigenschaften.
Welche Gruppennamen angegeben werden können, ist ausschließlich installationsabhängig und muss dokumentiert sein. Beispiel: UNIT=PLATTE
Oft definierte Gruppennamen sind:

- SYSDA alle oder bestimmte Magnetplattengeräte
- SYSSQ alle oder bestimmte Magnetplatten- und Magnetbandgeräte
- CASS alle oder bestimmte Magnetband-Kassettengeräte
- TAPE alle oder bestimmte Magnetband-Rollengeräte
- VIO virtuelle Magnetplattengeräte (Simulation im Hauptspeicher)

4.2.4 Schlüsselwortparameter: VOL bzw. VOLSER

Verwendung (vereinfachte Darstellung)

Bei Datenbeständen auf Platte oder Band können für das Anlegen oder das Wiederfinden ein oder mehrere Datenträgernamen [Volume Serials] (VOL bzw. VOLSER) angegeben werden.
Der VOL-Parameter kann beim Anlegen eines Datenbestandes entfallen, wenn der Ersteller keinen bestimmten Datenträger wünscht.
Der VOL-Parameter kann beim wieder Finden eines Datenbestandes entfallen, wenn ein “Katalogeintrag‘ dafür existiert. Der hier erwähnte ‘Katalogeintrag‘ wird in einem späteren Abschnitt ausführlich behandelt.
Der VOL-/VOLUME-Parameter wirkt mit dem UNIT-Parameter und mit dem SPACE-Parameter zusammen.

Parametertyp

Schlüsselwortparameter, optional.

VOL=SER=serialnum oder **VOL=SER=(serialnum,serialnum,...)**

Hiermit werden ein oder mehrere Datenträgernamen [Volume Serials] für den Datenbestand ausgewählt. Welche Datenträgernamen angegeben werden können, ist installationsabhängig und muss dokumentiert sein.

> Wird nur ein Datenträgername angegeben, so wird der Datenbestand ausschließlich auf diesem Datenträger gespeichert.
> Beispiel VOL=SER=ABC123

Wird eine Liste von Datenträgernamen angegeben, so wird der Datenbestand zuerst auf dem ersten Datenträger dieser Liste gespeichert. Bei Bedarf können weitere Teilbereiche dieses Datenbestandes auf den zusätzlichen, in dieser Liste spezifizierten Datenträgern gespeichert werden (Multi-Volume-Dataset).

> Beispiel VOL=SER=(ABC123,ABC456,ABC789)

Wird *kein* Datenträgername angegeben, so wählt das System einen Datenträger entsprechend dem UNIT-Parameter aus (Non-Specific-Volume-Request).
Bei Datenbeständen auf Platte wird die Größe der Teilbereiche durch den SPACE-Parameter beeinflusst.

4.2.5 Schlüsselwortparameter: SPACE (ohne SMS)

Verwendung (vereinfachte Darstellung)

Bei Datenbeständen auf Platte (nicht auf Band) muss beim Anlegen der Platzbedarf (SPACE) angegeben werden.
Der SPACE-Parameter wirkt mit dem UNIT-Parameter und mit dem VOL-/VOLUME-Parameter zusammen.

Parametertyp

Schlüsselwortparameter, optional.

SPACE=(TRK,(primary[,secondary])[,RLSE]) oder SPACE=(CYL,(primary[,secondary]>[,RLSE])

Mit dieser Form wird der Platzbedarf eines sequentiellen Datenbestandes festgelegt. Bei Einsatz von SMS kann auf diese Weise auch der Platzbedarf eines VSAM-Datenbestandes festgelegt werden.
(Alternative VSAM-Datenbestand mit Dienstprogramm IDCAMS anlegen.)

SPACE=(TRK,(primary,[secondary],dirblocks)[,RLSE]) oder SPACE=(CYL,(primary,[secondary],dirblocks)[, RLSE])

Mit dieser Form wird der Platzbedarf eines untergliederten Datenbestandes festgelegt.

SPACE-Einheit: TRK oder CYL

Hiermit wird festgelegt, ob der Platzbedarf in Einheiten von Spuren [Tracks] (TRK) oder in Einheiten von Zylindern [Cylinders] (CYL) angegeben wird.
Es ist notwendig, eine Vorstellung von der Kapazität einer Spur bzw. eines Zylinders zu haben. Bei Platten vom Typ 3390 (UNIT=3390) gilt:

- Kapazität einer Spur ca. 56 KB
- Kapazität eines Zylinders ca. 840 KB (15 Spuren/Zylinder)

RLSE

Die optionale Angabe RLSE bewirkt, dass nach einem erstmaligen Vorgang OPEN-Schreiben-CLOSE die noch unbenutzten Spuren bzw. Zylinder des Datenbestandes freigegeben werden (und damit durch andere Datenbestände in Besitz genommen werden können).
Diese Angabe ist vor allem dann sinnvoll, wenn ein Datenbestand bewusst überdimensioniert angelegt werden soll, die Menge der zu speichernden Daten vorab jedoch nicht bekannt ist.

primary

Hiermit wird die anfängliche Große (Primärmenge) des Datenbestandes in Spuren bzw. Zylindern definiert. In der Regel wird Platz in der angegebenen Größe zusammenhängend (d.h. als ein Extent) zugeordnet. Ist das nicht möglich, ordnet das System die Primärmenge in bis zu 5 Extents zu.
Ist mittels VOL=SER= ein bestimmter Datenträger spezifiziert, und ist dort nicht genügend freier Platz zur Aufnahme der Primärmenge, dann wird das Anlegen des Datenbestandes vom System verweigert.
Ist kein bestimmter Datenträger spezifiziert (Non-Specific-Volume-Request), so durchsucht das System alle aufgrund der UNIT-Angabe in Frage kommenden Datenträger, bis einer gefunden wird, auf dem genügend freier Platz zur Verfügung steht.

secondary

Hiermit wird optional eine Erweiterungsmenge (Sekundärmenge) des Datenbestandes in Spuren bzw. Zylindern definiert. Diese Sekundärmenge wird erst dann zugeordnet, wenn die bisherige Größe des Datenbestandes nicht ausreicht. In der Regel wird Platz in der angegebenen Größe zusammenhängend (d.h. als ein Extent) zugeordnet. Ist das nicht möglich, ordnet das System jede Sekundärmenge in bis zu 5 Bereichen zu.
Es können maximal 15 Sekundärmengen zugeordneten werden. Die Anzahl von insgesamt 16 Extents (1 Extent für die primäre Zuweisung und 15 Extents für die sekundäre Zuweisung) kann jedoch nicht überschritten werden. Bei PDS/E-Datasets beträgt die maximale Anzahl 123 Extents.
Sind mittels VOL=SER= mehrere Datenträger spezifiziert (Multi-Volume-Dataset), so können Erweiterungen des Datenbestandes auch auf den übrigen Datenträgern angelegt werden.
Wird eine Erweiterungsmenge nicht angegeben oder als 0 angegeben, dann ist eine Erweiterung der bisherigen Größe eines Datenbestandes nicht möglich.
Ein ‘Überlaufen‘ eines Datenbestandes führt in der Regel zu einem abnormalen Ende (ABEND).

dirblocks (nur bei untergliederten Datenbeständen)

Hiermit wird die Größe des Inhaltsverzeichnisses (der Directory) eines untergliederten Datenbestandes definiert.
Ein ‘dirblock‘ ist 256 Bytes groß. Wesentlich hilfreicher ist jedoch die Aussage, dass in einem ‘dirblock‘ mindestens 5 Elemente (Members), maximal 21 Elemente verwaltet werden können.
Um ‘auf der sicheren Seite‘ zu sein, ist es z.B. sinnvoll, zur Verwaltung von bis zu 50 Elementen eine Anzahl von 10 ‘dirblocks anzugeben.
Bei modernen untergliederten Datenbeständen vom Typ PDSE wird diese Angabe ignoriert.

4.2.6 Eigenschaften von Datenbeständen

Allgemein

Alle Datenbestände haben bestimmte Eigenschaften. Diese Eigenschaften werden systemintern durch einen Dateisteuerblock [Dataset Control Block] dargestellt und können zu einem gewissen Maße innerhalb einer DD-Anweisung ergänzt werden.
Die Eigenschaften werden durch bestimmte Schlüsselwörter und zugehörige Werte beschrieben.

DSORG=org

Die Dateiorganisation [Dataset Organization] kann sein:

- PS für sequentielle Organisation
  [Physical Sequential] (PS-Datenbestand)
- PO für untergliederte Organisation
  [Partitioned Organized] (PO-Datenbestand, Bibliothek, Library)

RECFM=recfm

Das Satzformat [Record Format] kann sein:

- F, FB für feste Satzlänge, ungeblockt bzw. geblockt
- V, VB für variable Satzlänge, ungeblockt bzw. geblockt
- U für undefinierte Satzlange bzw. unbekannte Satzstruktur
- xxA Zusatz ANSI-genormte Ausgabesteuerzeichen in Byte 1
- xxM Zusatz maschinenabhängige Ausgabesteuerzeichen in Byte 1

LRECL=rlen

Die Satzlänge [Logical Record Length] gibt an:

- die exakte Satzlänge (1...32760) bei RECFM=F/FB
- die maximale Satzlänge (1.. .32756) bei RECFM=V/VB

BLKSIZE=blen

Die Blockgröße [Blocksize] gibt an:

- die exakte Blockgröße (1 . .32760) bei RECFM=F/FB
- die maximale Blockgröße (1 .. .32760) bei RECFM=V/VB

Wird beim Anlegen eines Datenbestandes keine BLKSIZE angegeben, so legt das System einen vom Typ des Datenträgers abhängigen, optimalen Wert fest.

4.2.7 Schlüsselwortparameter: DCB

Verwendung (vereinfachte Darstellung)

Die Eigenschaften eines Datenbestandes können gemeinsam durch den DCB-Parameter und dessen Unterparameter beschrieben werden.

Parametertyp

Schlüsselwortparameter, optional.

DCB=([DSORG=org][,RECFM=recfm][,LRECL=rlen][,BLKSIZE=blen])

Hiermit werden Eigenschaften eines Datenbestandes beschrieben. Die Unterparameter entsprechen den oben beschriebenen Schlüsselwörtern.

4.2.8 Schlüsselwortparameter: RECFM, LRECL, BLKSIZE

Verwendung (vereinfachte Darstellung)

Die Eigenschaften eines Datenbestandes können alternativ (d.h. statt mit Hilfe des DCB-Parameters) einzeln durch die Parameter RECFM, LRECL und BLKSIZE beschrieben werden.

Parametertyp

Schlüsselwortparameter, optional.

RECFM=recfm bzw. **LRECL=rlen** bzw. **BLKSIZE=blen**

Hiermit werden Eigenschaften eines Datenbestandes beschrieben. Die Parameter entsprechen den oben beschriebenen Schlüsselwörtern.
Wenn der Parameter BLKSIZE weggelassen wird, dann wird die für den ausgewählten Datenträger optimale Blockgröße durch das SMS ergänzt.

4.2.9 Sonstige Quellen für DCB-Werte

Art der Quellen

Die tatsächlich wirksamen DCB-Werte können aus 3 Quellen stammen:

- aus dem auszuführenden Programm (in der Regel bei PGM= angegeben)
- aus der zugehörigen DD-Anweisung
- aus den Kennsätzen eines Datenbestandes auf Platte oder Band.

Vorrangregeln

Festlegungen im auszuführenden Programm haben absoluten Vorrang.
Werte, die im auszuführenden Programm nicht festgelegt werden (oder formal als 0 definiert sind), werden der DD-Anweisung entnommen, falls sie dort spezifiziert sind.
Werte, die dann immer noch nicht festgelegt sind (oder formal als 0 definiert sind), werden den Kennsätzen des Datenbestandes entnommen, falls der Datenbestand durch Kennsätze beschrieben ist und die noch fehlenden Werte dort spezifiziert sind.
Sind letztlich immer noch nicht alle Werte festgelegt, oder wurde eine widersprüchliche Kombination von Werten festgelegt (z.B. RECFM=FB, LRECL=100, BLKSIZE=32760), dann ist in der Regel ein abnormales Ende (ABEND) die Folge.
Eine Faustregel, ob und ggf. welche Werte in der DD-Anweisung spezifiziert werden sollen, kann nicht gegeben werden. Nur bei genauer Kenntnis (bzw. ausführlicher Dokumentation) des auszuführenden Programms kann entschieden werden, welche Werte in der DD-Anweisung angegeben werden sollten.

Hinweis

Datenbestände auf Platte sind immer durch Kennsätze (im VTOC der Platte) beschrieben. Datenbestände auf Band sind in der Regel durch Kennsätze (ebenfalls auf dem Band) beschrieben. Bänder ohne Kennsätze gelten als No-Label-Bänder.

4.2.10 Schlüsselwortparameter: LABEL

Verwendung (vereinfachte Darstellung)

Bei Datenbeständen auf Band (nicht auf Platte) kann es notwendig sein, einige Angaben über die Art der Abspeicherung des oder der Datenbestände auf dem Band zu spezifizieren.

Parametertyp

Schlüsselwortparameter, optional.

LABEL=([sequencenum][,labeltype])

Der LABEL-Parameter wird benötigt, wenn

- mehr als ein Datenbestand auf ein Band gespeichert werden soll
- oder dort schon gespeichert ist
- wenn auf dem Band entweder keine Kennsätze oder Nicht-Standard-Kennsätze geschrieben werden sollen oder dort existieren.

sequencenum

Hiermit wird die Datenbestands-Folgenummer [Dataset-Sequence-Number] spezifiziert. Wird dieser Wert beim Anlegen eines Datenbestandes nicht angegeben, dann wird dieser Datenbestand als erster oder einziger auf das Band geschrieben. Das ist gleichwertig einer Angabe von LABEL=1.
Um mehrere Datenbestände hintereinander auf ein Band schreiben zu können, *muss* dieser Wert angegeben werden. Beispiel: LABEL=2
Wurde beim Anlegen eines solchen Datenbestandes ein “Katalogeintrag erstellt, kann dieser Wert zum wieder Finden des Datenbestandes entfallen. Wurde kein “Katalogeintrag“ erstellt, *muss* dieser Wert auch zum wieder Finden des Datenbestandes angegeben werden. Der hier erwähnte “Katalogeintrag“ wird in einem späteren Abschnitt ausführlich behandelt.

labeltype

Hiermit wird der Typ der Kennsätze [Labels] spezifiziert.
Standardmäßig werden Standard-Kennsätze benutzt. In diesem Fall braucht dieser Wert nicht angegeben zu werden. Folgende Werte können spezifiziert werden:

• SL	Standard-Kennsätze	[Standard Labels], ist Default
• NSL	Nicht-Standard-Kennsätze	[Non Standard Labels]
• NL	keine Kennsätze	[No Labels]
• BLP	Kennsätze werden ignoriert, d.h. als Datensätze behandelt	[Bypass Label Processing]. Aus Sicherheitsgründen ist diese Form in vielen Installationen unzulässig.

4.2.11 Schlüsselwortparameter: EXPDT und RETPD

Verwendung (vereinfachte Darstellung)

Bei Datenbeständen auf Platte oder Band kann ein Verfallsdatum [Expiration Date] (EXPDT) oder eine Schutzfrist [Retention Period] (RETPD) angegeben werden. Die Schutzfrist wird, vom heutigen Tag an gerechnet, intern ebenfalls in ein Verfallsdatum umgerechnet.
Der Datenbestand kann erst ab dem Verfallsdatum mit normalen Mitteln gelöscht oder überschrieben werden.
Soll der Datenbestand ausnahmsweise doch vor Erreichen des Verfallsdatums gelöscht werden, ist das Dienstprogramm IDCAMS zu verwenden.

Parametertyp

Schlüsselwortparameter, optional.

EXPDT=yyddd oder EXPDT=yyyylddd

Hiermit wird ein Verfallsdatum festgelegt

- entweder in der Form Jahr zweistellig und laufender Tag des Jahres dreistellig.
  Beispiel EXPDT=97034 für 3.2.97
- oder in der Form Jahr vierstellig und laufender Tag des Jahres dreistellig.
  Beispiel: EXPDT=1 997/034 für 3.2.1997.

Die zweite Form ist auch für Verfallsdaten geeignet, die nach dem 31.12.1999 liegen.

RETPDnnnn

Hiermit wird eine Schutzfrist zwischen 0 und 9999 Tagen festgelegt.
Beispiel: RETPD=365

4.3 Katalogeintrag

Allgemein

Der Katalog dient dazu, für existierende Datenbestande diejenigen Informationen zu verwalten, die zum wieder Finden des Bestandes erforderlich sind.
“Katalogisieren‘ bedeutet, die Informationen über einen Datenbestand in den Katalog aufzunehmen. Hierzu dient die Angabe DISP=(. .. ,CATLG) in einer DD-Anweisung.
“Entkatalogisieren“ bedeutet, die Informationen über einen Datenbestand wieder aus dem Katalog zu entfernen hierzu dient die Angabe DISP=(... ,UNCATLG) in einer DD-Anweisung.
Wird beim Löschen eines katalogisierten Datenbestandes die Angabe DISP=(... , DELETE) in einer DD-Anweisung benutzt, dann beinhaltet dieser Vorgang nur dann auch das Entkatalogisieren, wenn der Bestand mit Hilfe des Katalogeintrages wieder gefunden wurde. Das ist dann der Fall, wenn die VOL=SER-Angabe in der betreffenden DD-Anweisung nicht geschrieben wurde.
Das Verwalten von Katalogeinträgen kann auch mit Hilfe des Dienstprogramms IDCAMS erfolgen.

Nicht-VSAM-Datenbestände können, VSAM-Datenbestände müssen katalogisiert sein. Jedoch werden auch Nicht-VSAM-Datenbestände in der Regel katalogisiert.
Nicht-VSAM-Datenbestände können beliebig oft katalogisiert und entkatalogisiert werden. Jedoch werden sie in der Regel beim Anlegen auch katalogisiert und erst beim Löschen wieder entkatalogisiert.
Datenbestände, die ‘SMS-managed‘ sind, müssen katalogisiert sein. Das Thema “SMS“ wird in einem späteren Abschnitt ausführlich behandelt.

Inhalt des Katalogs

Als Ordnungs- und Suchkriterium im Katalog dient der DSN eines Datenbestandes. Somit können nicht mehrere Datenbestände mit demselben DSN gleichzeitig katalogisiert sein.
Beim Katalogisieren eines Nicht-VSAM-Datenbestandes werden folgende Informationen in den Katalog eingetragen:

- DSN — Name des Datenbestandes
- UNIT — Datenträgertyp, auf dem der Datenbestand gespeichert ist
- VOL=SER — Name des oder der Datenträger, auf dem oder denen der Datenbestand gespeichert ist
- LABEL — Datenbestands-Folgenummer (bei Bestand auf Band)

Für VSAM- und ‘SMS-managed‘-Datenbestände werden wesentlich mehr Informationen in den Katalog eingetragen.

4.4 Weitere Anwendungen

4.4.1 Datenbestand kopieren mit IEBGENER (weitere Formen)

Programmaufruf

Programmname ist IEBGENER. Der Programmaufruf lautet:

```
//stepname  EXEC  PGM=IEBGENER
```

Datei-Beschreibungen

Das Programm IEBGENER benötigt 4 DD-Anweisungen.

SYSUT1

Beschreibung der Eingabedatei. Beliebiger sequentieller Datenbestand, z.B.:

```
//SYSUT1    DD DSN=FILIALE.HAMBURG.EINGABE,
//             DISP=SHR
```

Alternativ kann als Eingabedatei ein bestimmtes Element (Member) eines untergliederten Datenbestandes dienen, z.B.:

```
//SYSUT1    DD DSN=PROD.MVS.LIB(MEM1),
//          DISP=SHR
```

SYSUT2

Beschreibung der Ausgabedatei. Beliebiger sequentieller Datenbestand, z.B.:

```
//SYSUT2    DD DSN=FILIALE.HAMBURG.NONFOOD.UMSATZ,
//          DISP=(NEW,CATLG,DELETE),UNIT=SYSKD,
//          SPACE=(CYL,2,2)),RECFM=FB,LRECL=400
//
```

Alternativ kann als Ausgabedatei ein bestimmtes Element (Member) eines untergliederten Datenbestandes ersetzt oder neu erstellt werden, z.B.:

```
//SYSUT2    DD DSN=PROD.MVS.LIB(MEMX),
//             DISP=SHR
```

SYSPRINT

Beschreibung einer Protokolldatei für Nachrichten und Fehlermeldungen, als Druckausgabe gestaltet, z.B.:

```
//SYSPRINT  DD SYSOUT=*
```

SYSIN

Beschreibung einer Datei für Dienstprogramm-Steueranweisungen, im Lochkartenformat gestaltet (und somit definierbar als Instream-Datenbestand).
Wenn nur eine einfache 1:1-Kopieroperation durchgeführt werden soll, kann auf Dienstprogramm-Steueranweisungen verzichtet werden (und somit eine Definition als Scheindatei erfolgen), z.B.:

```
//SYSIN     DD DUMMY
```

5. Standard- und Dienstprogramme

5.1 Allgemeines

5.1.1 Überblick

Verwendung

Standardprogramme sind Programme, die vom Hersteller des Betriebssystems oder von unabhängigen Software-Anbietern in der Regel gegen eine Lizenzgebühr zur Nutzung angeboten werden, und mit denen bestimmte, häufig auszuführende Aufgaben bearbeitet werden können.
Dienstprogramme [Utilities] sind vom Hersteller des Betriebssystems zur Verfügung gestellte Programme, mit denen eine Vielzahl von Verwaltungs- und/oder Routinearbeiten gelöst werden kann.
Nachfolgend sind einige ausgewählte Standard- und Dienstprogramme aufgeführt und ihre Funktion erläutert.

IEFBR14	Dienstprogramm zum Durchführen elementarer Dateifunktionen. z.B.: Anlegen, Löschen von Datasets.
IEHLIST	Dienstprogramm zum Auflisten von System-Steuerdaten, z.B. VTOC- und Directory-Einträgen.
IEBGENER	Dienstprogramm zum Kopieren von sequentiellen Datenbeständen.
IEBCOPY	Dienstprogramm zum Verwalten, Kopieren, Laden und Entladen von untergliederten Datenbeständen.
IEBCOMPR	Dienstprogramm zum Vergleichen von sequentiellen und untergliederten Datenbeständen.
IDCAMS	Dienstprogramm zum Verwalten, Kopieren und Drucken von VSAM- und Nicht-VSAM-Datenbeständen.
SORT oder **ICEMAN**	Standardprogramm zum Sortieren, Mischen und Kopieren von sequentiellen und VSAM-Datenbeständen.

Weitere Informationen siehe Schulungsunterlagen zu dem Thema: z/OS Dienstprogramme.

6. Job-, Step-Steuerung

6.1 Allgemeines

6.1.1 Überblick

Verwendung

Unter Job-Steuerung versteht man die kontrollierte Aufeinanderfolge bzw. den kontrollierten Parallellauf mehrerer Jobs.

Unter Step-Steuerung versteht man die kontrollierte Aufeinanderfolge der einzelnen Steps eines Jobs.

Job-Steuerung

Zur Job-Steuerung existieren folgende Methoden:

- in einen MVS/JES2-System
  keine allgemein sinnvolle Methode.
- in einen MVS/JESS-System
  die Methode der Job-Netze oder auch DJC [Dependent Job Controll genannt, realisiert durch den Einsatz spezieller JES3-JCL-Anweisungen wie z.B.
  ```
  //*NET    NETID=...,NHOLD=...,RELEASE=...
  ```
- in einen MVS/JES2- oder MVS/JES3-System
  Einsatz eines spezifischen Job-Planungs- und Steuerungs-Systems, entwickelt vom Hersteller des Betriebssystems oder von unabhängigen Software-Anbietern.

Die interne Organisation dieser Installation und die gewählte Methode müssen gut aufeinander abgestimmt sein. Die Behandlung solcher Methoden fällt aus dem Rahmen dieses Seminars und wird hier nicht behandelt.

Step-Steuerung

Zur Step-Steuerung existieren folgende Methoden:

- Spezifizieren von Bedingung(en) zur vorzeitigen Beendigung eines Joblaufes (COND-Parameter der JOB-Anweisung)
- Spezifizieren von Bedingungen zum Unterdrücken oder Ausführen eines bestimmten Steps (COND-Parameter der EXEC-Anweisung)
- Spezifizieren von Bedingung(en) zum Unterdrücken oder Ausführen von Steps oder Stepfolgen (IF/THEN-, ELSE-, ENDIF-Anweisungen)

6.1.2 Regeln für die Step-Steuerung

Grundregeln

Ein Job kann aus maximal 255 Steps bestehen. Im Normalfall werden alle Steps eines Jobs nacheinander in der gegebenen Reihenfolge ausgeführt. Tritt jedoch in einem Step ein Abbruch (abnormales Ende (ABEND]) auf, dann werden standardmäßig alle nachfolgenden Steps unterdrückt.

Formelles Ergebnis eines Steps

Abweichungen von den Grundregeln können unter Bezugnahme auf das formelle Ergebnis eines Steps definiert werden:

- er läuft und kommt zu einem normalen Ende (kein Abbruch). Das ausgeführte Programm liefert einen Rückgabecode [Returncode, RC]
- er läuft und kommt zu einem abnormalen Ende (Abbruch [ABEND]). Es erscheint ein Abbruch-Code [ABEND-Code] der Form Sxxx oder Unnnn
- er läuft nicht, d.h. die Ausführung des Steps wird unterdrückt. Als formelles Ergebnis erscheint ‘NOT EXECUTED‘ bzw. ‘FLUSH’

Rückgabecode [Returncode, RC]

Der vom ausgeführten Programm erzeugte Rückgabecode [Returncode, RC] (ein Zahlenwert zwischen 0 und 4095) eines Steps kann als Kriterium für das Ausführen oder Unterdrücken von Folgesteps benutzt werden.

Abbruchcode [ABEND-Code]

Das abnormale Ende kann vom System erzwungen werden (System-ABEND) oder als freiwillige Maßnahme (User-ABEND, vergleichbar mit einer „Selbstaufgabe“) vom ausgeführten Programm veranlasst werden.
Beim System-ABEND liefert das System einen System-ABEND-Code der Form Sxxx (‘xxx‘ ist ein Hexadezimalwert zwischen ‘001‘ und ‘FFF). Die Bedeutung des Codes ist in der Betriebssystem-Literatur dokumentiert.
Beim User-ABEND liefert das ausgeführte Programm einen User-ABEND-Code der Form Unnnn (‘nnnn’ ist ein Dezimalwert zwischen 0001 und 4095), die Bedeutung des Codes muss in der Programm-Beschreibung dokumentiert sein.
Die Tatsache eines ABENDs in einem Step kann als Kriterium für das Ausführen oder Unterdrücken von Folgesteps benutzt werden. Der ABEND-Code selbst kann jedoch nur mit IF/THEN-, ELSE-, ENDIF-Anweisungen ausgewertet werden, nicht mit den CaND-Parametern.

‘NOT EXECUTED‘ bzw. ‘FLUSH‘

Ein FLUSH in einem Step kann als Kriterium für das Ausführen oder Unterdrücken von Folgesteps benutzt werden, jedoch nur mit IF/THEN-, ELSE-, ENDIF-Anweisungen, nicht mit den COND-Parametern.

6.2 COND-Parameter der JOB-Anweisung

6.2.1 Bedeutung und Syntax

Verwendung

Der COND-Parameter der JOB-Anweisung ermöglicht die Angabe einer oder mehrerer (max. 8) Bedingungen, bei deren Auftreten der Joblauf vorzeitig beendet werden soll.
Es wird ein Rückgabecode-Test [Returncode-Test, RC-Test] durchgeführt. Der Rückgabecode [Returncode, RC] ist ein Zahlenwert zwischen 0 und 4095. Der Test bezieht sich beim Beginn eines neuen Steps auf alle vorangegangenen Steps. Falls der Test bzw. einer der angegebenen Tests das Ergebnis WAHR [TRUE] liefert, wird der Joblauf vorzeitig beendet (d.h. alle nachfolgenden Steps werden unterdrückt).
Steps, die unterdrückt wurden, werden bei den Tests nicht berücksichtigt.

COND(code,op) oder **COND=((code,op)[,(code,op)]...)**

gibt die (max. 8) Bedingung(en) zur vorzeitigen Beendigung des Joblaufes an. Es genügt, dass *ein* Test erfüllt ist, um den Joblauf vorzeitig zu beenden

code

gibt einen zulässigen Zahlenwert für einen Returncode-Test an (zwischen 0 und 4095)

op

Vergleichsoperator:

EQ	gleich [equal]
NE	ungleich [not equal]
GT	größer als [greater than]
GE	großer als oder gleich [greater than or equal]
LT	kleiner als [less than]
LE	kleiner als oder gleich [less than or equal]

Hinweise

Der COND-Parameter der JOB-Anweisung hat Vorrang vor allen anderen Methoden zur Step-Steuerung. Wenn also wenigstens *einer* dieser Tests erfüllt ist, werden alle anderen Tests nicht weiter beachtet.
Die Logik dieser Tests ist nach einem unüblichen Schema aufgebaut.
Beispiel (12,LE) bedeutet

ist 12 *kleiner oder gleich* dem RC irgendeines Vorgängersteps?

üblicher

ist der RC irgendeines Vorgängersteps *größer oder gleich* 12?

6.3 COND-Parameter der EXEC-Anweisung

6.3.1 Bedeutung

Verwendung

Der COND-Parameter der EXEC-Anweisung ermöglicht die Angabe einer oder mehrerer Bedingungen, bei deren Auftreten der aktuelle Step unterdrückt oder ausgeführt werden soll.
Es wird durchgeführt:

- entweder ein Returncode-Test auf alle vorangegangenen Steps
- oder ein Returncode-Test auf einen bestimmten vorangegangenen Step
- oder ein ABEND-Test (‘EVEN oder ‘ONLY‘) auf alle vorangegangenen Steps
- oder eine Kombination (max. 8 Tests) hieraus.

Vorangegangene Steps, die unterdrückt wurden, werden bei den Tests *nicht* berücksichtigt.

Falls (wenigstens) einer der angegebenen Returncode-Tests das Ergebnis WAHR [TRUE] liefert, wird der aktuelle Step unterdrückt. Diese Returncode-Tests haben Vorrang vor einem evtl. ‘EVEN‘ oder ‘ONLY.
Falls kein Returncode-Test angegeben ist oder keiner der angegebenen Re - Tests das Ergebnis WAHR [TRUE] liefert, jedoch ‘EVEN angegeben ist, wird der aktuelle Step immer ausgeführt, auch wenn in einem vorangegangenen Step ein ABEND auftrat.
Falls kein Returncode-Test angegeben ist oder keiner der angegebenen Returncode-Tests das Ergebnis WAHR [TRUE] liefert, jedoch ‘ONLY‘ angegeben ist, wird der aktuelle Step nur dann ausgeführt, wenn in einem vorangegangenen Step ein ABEND auftrat.

Job-Abbruch

Bei bestimmten Arten von System-ABENDs ist eine Fortsetzung des Jobs, d.h. die Ausführung von nachfolgenden Steps, *generell* nicht möglich. Somit sind ‘EVEN‘ oder ‘ONLY‘ wirkungslos.
Hierzu gehören u.a. folgende Situationen bzw. ABEND-Codes:

- JCL Error Fehler in den JCL-Anweisungen
- 122, 222 Job-Abbruch durch Operator
- 322 Überschreiten des CPU-Zeitlimits für den Job (vgl. TIME-Parameter der JOB-Anweisung)
- 522 Überschreiten des Wait-State-Limit
- 722 Überschreiten des Limits für die Druckausgabe

6.3.2 Syntax

COND=(code,op[,stepname])
COND=EVEN
COND=ONLY
COND=((code,op[,stepname])[,(code,op[,stepname])]...[,EVEN])
COND((code,op[,stepname])[,(code,op[,stepname])]...[,ONLY])
gibt die (max. 8) Bedingung(en) zum Unterdrücken oder Ausführen des aktuellen Steps an.

code

gibt einen zulässigen Zahlenwert für einen Returncode-Test an (zwischen 0 und 4095)

op

Vergleichsoperator:

EQ	gleich [equal]
NE	ungleich [not equal]
GT	größer als [greater than]
GE	großer als oder gleich [greater than or equal]
LT	kleiner als [less than]
LE	kleiner als oder gleich [less than or equal]

stepname

gibt den Namen eines bestimmten vorangegangenen Steps an. Wenn der Stepname angegeben ist, wird der Test für *diesen* vorangegangenen Step ausgeführt. Wenn der Stepname nicht angegeben ist, wird der Test für *alle* vorangegangenen Steps ausgeführt.

Hinweise

Der COND-Parameter der JOB-Anweisung hat Vorrang vor dem COND-Parameter der EXEC-Anweisung für den aktuellen Step.
Der aktuelle Step kann Teil einer Struktur sein, die mittels IF/THEN-, ELSE-, ENDIF-Anweisungen gesteuert wird. Somit erhält der COND-Parameter der EXEC-Anweisung die niedrigste Rangstufe.
Die Logik dieser Tests ist nach einem unüblichen Schema aufgebaut.

Beispiel: (12,LE,ST01) bedeutet:
ist 12 kleiner oder gleich dem RC des Vorgängersteps ST01?
üblicher: ist der RC des Vorgängersteps STO1 größer oder gleich 12?

6.4 IF/THEN-, ELSE-, ENDIF-Anweisungen

6.4.1 Überblick

Verwendung

Die lF/THEN-, ELSE-, ENDIF-Anweisungen dienen zur Steuerung der Steps oder Stepfolgen eines Jobs. Es werden eine oder mehrere Bedingungen angegeben, bei deren Auftreten bestimmte Steps ausgeführt oder Step unterdrückt werden sollen. Hierzu dient eine ähnliche IF-THEN-ELSE-Logik wie aus den meisten Programmiersprachen bekannt.
Der Formalismus besteht aus drei JCL-Anweisungen:

- IF-Anweisung. Leitet eine Steuerungs-Struktur ein. Beinhaltet die zu prüfende(n) Bedingung(en) und die abschließende Zeichenfolge ‘THEN, die den Wahr-Zweig einleitet.
- ELSE-Anweisung. Optional. Setzt eine Steuerungs-Struktur fort und leitet den Falsch-Zweig ein.
- ENDIF-Anweisung. Schließt eine Steuerungs-Struktur ab.

Es können bis zu 15 solcher Strukturen verschachtelt werden.

Namensfeld

Das Namensfeld ist optional. Der angegebene Name ist frei wählbar. Es ist jedoch sinnvoll, insbesondere bei verschachtelten Strukturen, die Zusammenhänge zwischen den Strukturelementen durch geeignet gewählte Namen deutlich zu machen.
Spalte 1 und 2 enthalten ’//‘. Der Name muss in Spalte 3 beginnen. Er darf maximal 8 Zeichen (Stellen) lang sein.
Das erste Zeichen muss alphabetisch (A. . .Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. ..Z, 0.. .9) oder eines der Sonderzeichen § # $ sein.
Dem Namen muss mindestens ein Blank folgen.

Operationsfeld

Der Operationscode muss IF bzw. ELSE bzw. ENDIF lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt. Dem Operationscode muss mindestens ein Blank folgen.

Parameterfeld

Nur das Parameterfeld der lF-Anweisung enthält Parameter in Form der zu prüfende(n) Bedingung(en). Die ELSE- und die ENDIF-Anweisungen enthalten keine Parameter.

6.4.2 Schlüsselwort-Bedingung: RC

Verwendung

Die Schlüsselwort-Bedingung RC ermöglicht:

- entweder einen Returncode-Test auf *keinen bestimmten* vorangegangenen Step. Der Test bezieht sich in diesem Fall auf den höchsten bisher aufgetretenen Returncode!!!
- oder einen Returncode-Test auf einen *bestimmten* vorangegangenen Step.

  RC op code oder **stepname.RC op code**

  gibt Returncode-Tests an.

stepname

Gibt den Namen eines bestimmten vorangegangenen Steps an. Wenn angegeben, wird der Test für *diesen* vorangegangenen Step ausgeführt. Ansonsten wird der Test für den höchsten bisher aufgetretenen Returncode ausgeführt. Bei Jobbeginn wird der Anfangs-Returncode auf 0 gesetzt.

op

Vergleichsoperator (Buchstabenkürzel oder Symboldarstellung zulässig):

EQ	=	gleich [equal]
NE	^=	ungleich [not equal]
GT	>	größer als [greater than]
GE	>=	größer als oder gleich [greater than or equal]
LT	<	kleiner als [less than]
LE	<=	kleiner als oder gleich [less than or equal]
NG	^>	nicht größer als [not greater than] - (wirkt wie LE)
NL	^<	nicht kleiner als [not less than] - (wirkt wie GE)

code

gibt einen zulässigen Zahlenwert für einen Returncode-Test an (zwischen 0 und 4095).

6.4.3 Schlüsselwort-Bedingung: ABENDCC

Verwendung

Die Schlüsselwort-Bedingung ABENDCC ermöglicht:

- entweder einen ABEND-Code-Test auf keinen bestimmten vorangegangenen Step. Der Test bezieht sich in diesem Fall auf zuletzt aufgetretenen ABEND-Code!!!
- oder einen ABEND-Code-Test auf einen bestimmten vorangegangenen Step.

ABENDCC op code oder **stepname.ABENDCC op code**
gibt ABEND-Code-Tests an.

stepname

gibt den Namen eines bestimmten vorangegangenen Steps an. Wenn angegeben, wird der Test für *diesen* vorangegangenen Step ausgeführt. Ansonsten wird der Test für den *zuletzt* aufgetretenen ABEND-Code ausgeführt.

op

Vergleichsoperator (Buchstabenkürzel oder Symboldarstellung zulässig):

EQ	=	gleich [equal]
NE	^=	ungleich [not equal]
GT	>	größer als [greater than]
GE	>=	größer als oder gleich [greater than or equal]
LT	<	kleiner als [less than]
LE	<=	kleiner als oder gleich [less than or equal]
NG	^>	nicht größer als [not greater than] - (wirkt wie LE)
NL	^<	nicht kleiner als [not less than] - (wirkt wie GE)

code

gibt einen zulässigen Wert an für einen

- System-ABEND-Code in der Form Sxxx (xxx‘ ist ein Hexadezimalwert zwischen ’001‘ und ‘FFF‘).
- User-ABEND-Code in der Form Unnnn (‘nnnn‘ ist ein Dezimalwert zwischen 0001 und 4095).

6.4.4 Schlüsselwort-Bedingung: ABEND

Verwendung

Die Schlüsselwort-Bedingung ABEND ermöglicht

- entweder einen ABEND-Test auf *keinen* bestimmten vorangegangenen Step. Der Test bezieht sich dann auf *alle* vorangegangenen Steps
- oder einen ABEND-Test auf einen *bestimmten* vorangegangenen Step.

ABEND=TRUE oder **stepname.ABEND=TRUE** oder
ABEND oder **stepname.ABEND**

gibt Tests an für das *Auftreten* eines ABEND in den vorangegangenen Steps.

ABEND=FALSE oder **stepname.ABEND=FALSE** oder
NOT ABEND oder **NOT stepname.ABEND** oder
^ABEND oder **^stepname.ABEND**

gibt Tests an für das Nicht-Auftreten eines ABEND in den vorangegangenen Steps.

stepname

Gibt den Namen eines bestimmten vorangegangenen Steps an. Wenn angegeben, wird der Test für diesen vorangegangenen Step ausgeführt. Ansonsten wird der Test für alle vorangegangenen Steps ausgeführt.

6.4.5 Schlüsselwort-Bedingung: RUN

Verwendung

Die Schlüsselwort-Bedingung RUN ermöglicht einen Test, ob ein bestimmter vorangegangener Step ausgeführt wurde oder unterdrückt wurde.

stepname.RUNTRUE oder **stepname.RUN**

prüft das Ausführen eines vorangegangenen Steps.

stepname.RUNFALSE oder **NOT stepname.RUN**
oder **^stepname.RUN**

prüft das Unterdrücken eines vorangegangenen Steps.

stepname

gibt den Namen des vorangegangenen Steps an, für den der Test ausgeführt werden soll.

6.4.6 Weitere Betrachtungen

Allgemeine Regeln

Vorangegangene Steps, die unterdrückt wurden, werden bei allen Tests (mit Ausnahme des RUN-Tests) nicht berücksichtigt.
Ein bestimmter Step innerhalb einer IF/THEN-, ELSE-, ENDIF-Struktur *kann* einen COND-Parameter in der EXEC-Anweisung enthalten. Dieser COND-Parameter wird *zusätzlich* nach den dafür gültigen Regeln überprüft und kann ggf. zur Unterdrückung des betroffenen Steps führen.

Job-Abbruch

Bei bestimmten Arten von System-ABENDs ist eine Fortsetzung des Jobs, d.h. die Ausführung von nachfolgenden Steps, generell nicht möglich. Somit sind die Bedingungen ‘ABEND‘ oder ABENDCC‘ wirkungslos.
Hierzu gehören die gleichen Situationen bzw. ABEND-Codes, die bereits beim COND-Parameter der EXEC-Anweisung aufgezeigt wurden.

Sonderregel zum 1. Step

Eine IF-Anweisung sollte **nicht vor** dem 1. Step in einem Job stehen. Falls doch, wird sie trotzdem erst **nach** Ausführung des 1. Steps ausgewertet!

Kombination von Bedingungen

Mehrere Bedingungen, auch verschiedenartige Schlüsselwort-Bedingungen, können zu einer komplexen Gesamt-Bedingung wie folgt kombiniert werden:

- Klammerung — Priorität 1 (höchste Priorität)
- NOT — bzw. ^ (logisches NICHT) — Priorität 2
- AND — bzw. & (logisches UND) — Priorität 3
- OR — bzw. | (logisches ODER) — Priorität 4

Innerhalb *einer* Prioritätsstufe werden die Einzelbedingungen von links nach rechts abgearbeitet. Das Ergebnis jeder Einzelbedingung ist WAHR [TRUE] oder FALSCH [FALSE]. Die Kombination liefert ein Gesamtergebnis WAHR oder FALSCH.
Das Gesamtergebnis entscheidet, ob die Steps des Wahr-Zweiges oder (falls definiert) des Falsch-Zweiges dieser IF/THEN-, ELSE-, ENDIF-Struktur ausgeführt werden.
Ist das Gesamtergebnis FALSCH, aber kein Falsch-Zweig definiert, so wird innerhalb dieser IFITHEN-, ELSE-, ENDIF-Struktur kein Step ausgeführt.

Schreibregeln

Eine komplexe Bedingung kann über mehrere Zeilen hinweg geschrieben werden, indem die Bedingung an jeder beliebigen Leerstelle unterbrochen und mit einer üblichen JCL-Folgezeile fortgesetzt wird.

6.5 Bezugnahmen und Rückbezugnahmen

6.5.1 Überblick

Verwendung

In bestimmten JCL-Anweisungen können einzelne Parameter durch Rückbezug auf einen bereits vorhandenen Parameter definiert werden:

- Format: **dsname**
  Rückbezugnamen zu existierenden katalogisierten Datenbeständen
- Format: ***.ddname**
  Rückbezugnahmen zu vorangehenden JCL-Anweisungen desselben Steps
- Format: ***.stepname.ddname**
  Rückbezugnahmen zu JCL-Anweisungen eines vorangehenden Steps.
- Format: ***.stepname.procstepname.ddname**
  Rückbezugnahmen zu JCL-Anweisungen eines Steps innerhalb eines vorangegangenen Prozeduraufrufs.

Damit soll gekennzeichnet werden, dass ein existierendes Objekt als Muster für ein neu zu erstellendes Objekt dienen soll, oder dass in einer späteren Si der gleiche Wert oder das gleiche Objekt benutzt werden soll wie in einer früheren Situation.

6.5.2 Bezugnahmen

Bezugnahmen bei DD-Anweisungen

Bezugnahmen bei DD-Anweisungen können verwendet werden bei

- dem VOL-/VOLUME-Parameter in der Form VOL=REF=
- dem DCB-Parameter
- dem LIKE-Parameter (nur bei Einsatz von SMS)

VOL=REF=dsname

Der Bezug betrifft den oder die Datenträgernamen [Volume Serials], auf denen der angegebene Datenbestand gespeichert ist.

DCB=dsname

Der Bezug betrifft

- die DCB-Werte, die der angegebene Datenbestand (auf Platte) aufweist. Ein Abändern einzelner DCB-Werte ist möglich, indem die abgeänderten Werte zusätzlich spezifiziert werden.
- das Verfallsdatum des angegebenen Datenbestandes

LIKE=dsname

Nur bei Einsatz von SMS. SMS und der LIKE-Parameter werden in einem späteren Abschnitt behandelt.

6.5.3 Rückbezugnahmen

Rückbezugnahmen bei DD-Anweisungen

Rückbezugnahmen bei DD-Anweisungen können verwendet werden bei:

- dem DSN-Parameter
- dem VOL-/VOLUME-Parameter in der Form VOL=REF=
- dem DCB-Parameter
- dem REFDD-Parameter (nur bei Einsatz von SMS)
- dem OUTPUT-Parameter

DSN=*.ddname oder **DSN=*.stepname.ddname** oder **DSN=*.stepname.procstepname.ddname**

Der Rückbezug betrifft

- den DSN aus dem angegebenen Step (einschließlich Membername), auch wenn darin der DSN-Parameter fehlt (temporärer Datenbestand).

VOL=REF*.ddname oder **VOL=REF=*.stepname.ddname** oder **VOL=REF=*.stepname.procstepname.ddname**

Der Rückbezug betrifft

- den Datenträgernamen [Volume Serials] aus dem angegebenen Step, auch wenn darin VOL=SER= weggelassen wurde
- den Typ der Kennsätze [Labels] aus dem LABEL-Parameter des angegebenen Steps.
- Bei der Angabe “dsname“ erfolgt der Bezug auf einen bereits katalogisierten Datenbestand.

DCB=*.ddname oder **DCB=*.stepname.ddname** oder **DCB=*.stepname.procstepname.ddname**

Der Rückbezug betrifft nur die tatsächlich spezifizierten DCB-Werte aus dem angegebenen Step. Ein Abändern einzelner DCB-Werte ist möglich, indem die abzuänderten Werte zusätzlich spezifiziert werden.

REFDD=*.ddname oder **REFDD*.stepname.ddname** oder **REFDD=*.stepname.procstepname.ddname**

Der Rückbezug gilt nur bei Einsatz von SMS. SMS und der REFDD-Parameter werden in einem späteren Abschnitt behandelt.

OUTPUT*.outname oder **OUTPUT *.stepname.outname** oder **OUTPUT =*.stepname.procstepname.outname**

Der Rückbezug betrifft OUTPUT-Anweisungen. Diese Anweisungen werden in einem späteren Abschnitt behandelt.

7. Datenbestandsbeschreibung (3. Teil)

7.1 Reservierte DD-Namen

7.1.1 Überblick

Verwendung

Bestimmte Namen in DD-Anweisungen („DD-Namen‘) sind für bestimmte Zwecke reserviert und können daher nicht frei benutzt werden. Reserviert sind u.a. folgende DD-Namen:

• SYSMDUMP	zur Ausgabe eines Speicherdrucks (Dump)
• SYSABEND • SYSUDUMP	zur Ausgabe eines formatierten Dump
• SYSCHK • SYSCKEOV	zum Erstellen eines Checkpoint-Datenbestandes
• JOBCAT • STEPCAT	zur Angabe eines bestimmten Katalogs
• JOBLIB • STEPLIB	zur Angabe von Lademodul-Bibliotheken

In der Regel werden nur folgende DD-Namen benutzt:

• SYSABEND	zur Auswertung des Dumps sind ASSEMBLER-Kenntnisse erforderlich
• SYSUDUMP	
• JOBLIB • STEPLIB	werden nachfolgend behandelt

7.1.2 JOBLIB, STEPLIB

Verwendung

JOBLIB und/oder STEPLIB DD-Anweisungen werden verwendet, wenn die auszuführenden Programme in einer oder mehreren bestimmten Lademodul-Bibliothek gesucht werden sollen.
JOBLIB und/oder STEPLIB DD-Anweisungen können häufig entfallen. Sie werden nur dann benötigt, wenn

- die auszuführenden Programme nicht in einer der standardmäßig festgelegten Lademodul-Bibliotheken liegen. Letztere sind in jeder Installation in der LINKLIST definiert.
- namensgleiche Programme in verschiedenen Lademodul-Bibliotheken existieren und gezielt ein bestimmtes Programm aus einer bestimmten Lademodul-Bibliothek ausgeführt werden soll.

JOBLIB

gilt für die Dauer eines Jobs. Eine JOBLIB DD-Anweisung muss, falls verwendet,

- nach der JOB-Anweisung und
- vor der ersten EXEC-Anweisung

geschrieben werden.

STEPLIB

gilt für die Dauer eines Steps. Eine STEPLIB DD-Anweisung muss, falls verwendet, nach der zu diesem Step gehörenden EXEC-Anweisung geschrieben werden.
Bei Steps, in denen eine STEPLIB definiert ist, wird

- zuerst in der durch STEPLIB definierten Lademodul-Bibliothek gesucht
- falls das auszuführenden Programm dort nicht gefunden wird, wird zusätzlich in der LINKLIST nach diesem Programm gesucht.

Sofern eine JOBLIB definiert ist, bleibt sie für *diesen* Step außer Betracht.

Bei Steps, in denen keine STEPLIB definiert ist, wird

- zuerst in der durch JOBLIB definierten Lademodul-Bibliothek gesucht, wenn im Job eine JOBLIB DD-Anweisung existiert
- falls keine JOBLIB definiert ist oder das auszuführenden Programm dort nicht gefunden wird, wird zusätzlich in der LINKLIST nach diesem Programm gesucht.

Hinweise zu Lademodul-Bibliotheken

Lademodul-Bibliotheken werden separat behandelt. Mehrere Lademodul-Bibliotheken können bei JOBLIB und/oder STEPLIB DD-Anweisungen in Form einer Verkettung angegeben werden. Auch Verkettungen werden separat behandelt.

7.2 Verketten von DD-Anweisungen

7.2.1 Überblick

Verwendung

Bei einer Verkettung [Concatenation] werden mehrere DD-Anweisungen (und damit mehrere physische Datenbestände) als ein logischer Datenbestand unter einem DD-Namen zusammengefasst.
Eine Verkettung kann nur für Eingabe-Datenbestände, nie für Ausgabe-Daten bestände spezifiziert werden.

Regeln

Die physischen Datenbestände innerhalb einer Verkettung müssen folgende physische Gemeinsamkeiten aufweisen:

- Alle Datenbestände müssen die gleiche DSORG aufweisen, wobei entweder DSORG=PS *oder* DSORG=PO zulässig ist. PS und PO können dann verkettet werden, wenn bei PO ein Member angegeben ist. Das Member einer PO entspricht einer PS-Datei.
- Alle Datenbestände müssen die gleiche RECFM aufweisen. Zulässig ist RECFM=Fxx *oder* RECFM=Vxx *oder* RECFM=Uxx, aber keine Mischung daraus.
- Bei RECFM=Fxx müssen alle Datenbestände die gleiche LRECL aufweisen bzw. dürfen bei RECFMVxx die LRECLs der weiteren Datenbestände nicht größer sein als die des ersten Bestandes der Verkettung. Diese Forderung kann ggf. durch Angabe einer ‘künstlichen‘ großen LRECL bei der ersten verketteten DD-Anweisung immer erfüllt werden.
- In der Regel dürfen die BLKSIZEs der weiteren Datenbestände nicht größer sein als die des ersten Bestandes der Verkettung. Diese Forderung kann ggf. durch Angabe einer ‘künstlichen‘ großen BLKSIZE bei der ersten verketteten DD-Anweisung immer erfüllt werden.

Schema

```
//ddname   DD <1.Datenbestand der Verkettung>,
//            [BLKSIZE=<Maximalwert>,]
//            [LRECL=<Maximalwert>]
//         DD <2.Datenbestand der Verkettung>
//         DD <n.Datenbestand der Verkettung>
```

7.2.2 Einsatz

Verkettung sequentieller Datenbestände

Die Verkettung mehrerer sequentieller Datenbestände erscheint dem Anwendungsprogramm wie ein einziger sequentieller Eingabe-Datenbestand.

- Die Situation ‘Ende der Eingabedatei tritt erst nach der Verarbeitung aller verketteter Datenbestände auf.
- Es können maximal 255 sequentielle Datenbestände verkettet werden.
- Ein Element eines untergliederten Datenbestandes gilt in diesem Zusammenhang wie ein sequentieller Datenbestand.

Nahezu jedes Anwendungsprogramm, das einen sequentiellen Eingabe-Datenbestand verarbeiten kann, kann statt dessen eine Verkettung mehrerer sequentieller Datenbestände verarbeiten. Auch z.B. die Programme IEBGENER (bei SYSUT1) und ICEMAN (bei SORTIN) können mit solchen Verkettungen arbeiten.

Verkettung untergliederter Datenbestände

Die Verkettung mehrerer untergliederter Datenbestände (Bibliotheken) erscheint dem Anwendungsprogramm wie eine (logische Gesamt-) Bibliothek mit mehreren Ebenen.
Bei der Benutzung einer solchen Verkettung soll immer ein bestimmtes Bibliotheks-Element gesucht und gefunden werden.
Die Suche nach einem bestimmten Element beginnt in der ersten Ebene der Bibliothek. Wird es hier gefunden, war die Suche erfolgreich und wird beendet.
War die Suche in der ersten Ebene erfolglos, wird in der nächsten Ebene weiter gesucht. Die Suche wird solange in jeweils der nächsten Ebene fortgesetzt, bis das gesuchte Element gefunden wird.
War die Suche auch in der letzten Ebene erfolglos, so war die Suche insgesamt erfolglos und wird beendet.
Es können unterschiedlich viele untergliederte Datenbestände (PDS und/oder PDSE) verkettet werden. Die Anzahl der Teilbereiche (Extents) darf 123 nicht überschreiten.
Diese Art der Verkettung wird vor allem bei Lademodul-Bibliotheken (STEPLIB, JOBLIB), Prozedur-Bibliotheken, Makro- und Copy-Bibliotheken benutzt.

7.3 Generations-Datenbestände

7.3.1 Überblick

Verwendung

Oft werden Datenbestände in regelmäßigen oder unregelmäßigen zeitlichen Abständen neu erstellt, obwohl sie dem Grunde nach dieselben Daten beinhalten. Unterscheidbar müssen diese Bestände jedoch im DSN sein. So erhält beispielsweise jeder neue Datenbestandes des jeweiligen Monats einen eigenen DSN.

3. Monat: DSN=LAGER.BESTAND.MONAT03
4. Monat: DSN=LAGER.BESTAND.MONAT04

Nachteil dieses Verfahrens ist u.a. die Notwendigkeit, in der JCL laufend den DSN zu aktualisieren.

Eine Definition eines solchen Bestandes als Generations-Datenbestand (GDS) [Generation Dataset] erleichtert die Gestaltung der JCL wesentlich. Ein GDS wird beim DSN-Parameter in der Regel mit Hilfe einer relativen Generationsnummer angesprochen.

Relative Generationsnummern

(0) neueste (aktuellste) Generation
(-1) vorherige (vorletzte) Generation
(-n) n-te vorherige Generation
(+1) in diesem Job neu zu erstellende (allerneueste) Generation. Diese muss katalogisiert werden, z.B. DISP=(NEW,CATLG).

Hinweis

Die relativen Generationsnummern bleiben für die Dauer eines Jobs unverändert. Erst nach Jobende verschieben sich diese Nummern d.h. aus dem bisherigen (-1) wird (-2), aus (0) wird (-1), aus (+1) wird (0).
Beispiel:
Erstellen jeweils eines neuen Datenbestandes mit Umsätzen am Ende der aktuellen Woche: DSN=LAGER.BESTAND(+1)

Absolute Generationsbezeichnungen

Die relativen Generationsnummern werden systemintern in absolute Generationsbezeichnungen der Form ‘GnnnnV00‘ (G: Generation, V: Version) umgesetzt und an den „Stamm“ des DSN angehängt.
Beispiel:

relativ: DSN=LAGER.BESTAND(+1)
absolut: DSN=LAGER.BESTAND.G3459V00

Die absoluten Generationsnummern werden, mit ‘0001‘ beginnend, fortlaufend hoch gezählt. Die Versionsnummer ist in der Regel ‘00‘.
Es ist erlaubt, beim DSN-Parameter auch den ‘richtigen‘ (d.h. den absoluten) Namen anzugeben.

7.3.2 Generations-Datengruppe (GDG)

Definition einer Generations-Datengruppe (GDG)

Zur Verwaltung von Generations-Datenbeständen muss eine Generations-Datengruppe (GDG) [Generation Data Group] eingerichtet werden. Hierzu nutzt man das Dienstprogramms IDCAMS („Access Method Services“) mit der Funktion DEFINE GENERATIONDATAGROUP oder DEFINE GDG.
Es sind der Name der GDG (‘NAME‘) sowie die Anzahl der zu verwaltenden Generationen (‘LIMIT‘ zwischen 1 und 255) zu spezifizieren.
Der Name eines Generations-Datenbestandes (GDS) ergibt sich dann ‘automatisch‘ aus dem Namen des GDG, ergänzt um die Angabe ‘GnnnnV00‘, wobei ‘nnnn‘ durch die nächste freie Generationsnummer ersetzt wird.
Beispiel:

GDG-Name: DSN=LAGER.BESTAND
GDS-Name: DSN=LAGER.BESTAND.G0001V00

In die Verwaltung als GDS innerhalb einer GDG können nur sequentielle (DSORG=PS) oder untergliederte (DSORG=PO) Datenbestände genommen werden.

EMPTY / NOEMPTY

Der optionale Zusatz EMPTY bzw. NOEMPTY gibt an, welche Maßnahme beim Erreichen des LIMIT durchgeführt wird:

EMPTY	alle bisher erstellten Generationen werden aus der Verwaltung genommen, d.h. entkatalogisiert
NOEMPTY	nur die älteste der bisher erstellten Generationen wird aus der Verwaltung genommen, d.h. entkatalogisiert (Default)

SCRATCH / NOSCRATCH

Der optionale Zusatz SCRATCH bzw. NOSCRATCH gibt an, welche Maßnahme für einen aus der Verwaltung fallenden GDS durchgeführt wird:

SCRATCH	Entkatalogisieren und Löschen des Bestandes
NOSCRATCH	nur Entkatalogisieren des Bestandes. Dieser bleibt als nicht katalogisierter Bestand physisch weiter bestehen (Default)

Die Default-Maßnahme NOSCRATCH ist bei Beständen auf Band sinnvoll, da Bänder in der Regel nicht physisch gelöscht werden. Bei Beständen auf Platte ist die Angabe SCRATCH daher zu empfehlen.

Syntax der Funktion DEFINE GDG

```
DEFINE GDG (NAME(gdgname) LIMIT(zahl) -
           [EMPTY|NOEMPTY] [SCRATCH|NOSCRATCH])
```

7.3.3 Generations-Datenbestand (GDS)

Zweck eines Modell-Datenbestandes

Bevor tatsächlich der erste Generations-Datenbestand (GDS) angelegt werden kann, muss ein Modell-Datenbestand (auch Modell-Datenbestandskenn-satz [Model-DSCB] genannt) existieren.
Für diesen Zweck kann prinzipiell jeder Datenbestand dienen. In der Regel wird jedoch in einer Installation *einmal* je ein Modell für sequentielle Datenbestände definiert. Für diese Modelle genügt ein Platzbedarf von 0 Spuren(!).

Definition eines Modell-Datenbestandes

```
//stepn   EXEC PGM=IEFBR14
//MODELPS DD   DSN=MODELL.GDGPS,DISP=(NEW,CATLG),
//             UNIT=SYSDA,SPACE=(TRK,0)
//             DCB=IRECFM=FB,LRECL=120)
```

Die Definition eines ModeII-Datenbestandes kann jedoch entfallen, wenn eine ‘SMS-managed-Generations-Datengruppe (SMS-GDG) benutzt wird.

Regeln zum Anlegen eines Generations-Datenbestandes (GDS)

Beim Anlegen eines Generations-Datenbestandes (GDS) ist anzugeben:

- in der Regel DISP(NEW,CATLG[,DELETE])
- als DSN der Name der GDG zusammen mit der relativen Generations-nummer (+1)
- geeignete UNIT- [und VOL=SER-] und/oder SPACE- und/oder LABEL-Angaben
- bei DCB den Bezug auf den Modell-Datenbestand. Die Eigenschaften des Modells können geändert werden!

Anlegen eines Generations-Datenbestandes (GDS)

```
//stepname EXEC PGM=IEFBR14
//GDS      DD  DSN=<gdgname>(+1),
//             DISP=(NEW,CATLG[,DELETE]),UNIT=<unit>,
//             SPACE=<space>,DCB=<modelldsname>
```

7.3.4 Weitere Möglichkeiten

Benutzen eines Generations-Datenbestandes (GDS)

Um einen Generations-Datenbestand (GDS) wieder zu benutzen, ist die relative Generationsnummer anzugeben.
Beispiel für die Verarbeitung der aktuellen bzw. vorherigen Generation:

```
//stepname EXEC PGM=<programm>
//GDSAKT   DD   DSN=<gdgname>(O ,DISP=(OLD,KEEP)
//GDSVORH  DD   DSN=<gdgname>(-1),DISP=(OLD,KEEP)
```

Benutzen aller Generations-Daten bestände

Um alle Generations-Datenbestände als einen Datenbestand zu verarbeiten, ist die relative Generationsnummer wegzulassen. Diese Art von Verarbeitung wirkt wie eine Verkettung aller existierender Generationen dieser GDG.
Beispiel für die Verarbeitung aller Generationen:

```
//stepname EXEC PGM=<prograrnrn>
//GDS      DD   DSN=<gdgname>,DISP=(OLD,KEEP)
```

Ändern der Eigenschaften einer Generations-Datengruppe (GDG)

Zur Änderung der Eigenschaften einer Generations-Datengruppe (GDG) dient das Dienstprogramms IDCAMS („Access Method Services“) mit der Funktion ALTER. Es sind der Name der GDG (‘NAME‘) sowie die geänderte Anzahl der zu verwaltenden Generationen (‘LIMIT‘ zwischen 1 und 255) zu spezifizieren.

Syntax der Funktion ALTER (mit LIMIT)

```
ALTER   gdgname LIMII(zahl)
```

Löschen einer Generations-Datengruppe (GDG)

Zum Löschen einer Generations-Datengruppe (GDG) dient das Dienstprogramms IDCAMS („Access Method Services“) mit der Funktion DELETE und dem Zusatz GDG. Vor dem Löschen der GDG sind alle zugehörigen Generations-Datenbestände zu löschen.

Syntax der Funktion DELETE

```
DELETE  (gdgname [...]) GDG
```

Syntax ohne Löschen der Generationen vorab

```
DELETE  (gdgname [...]) GDG RECOVERY
!!!!!!
```

7.4 OUTPUT-Anweisung zur Steuerung der Druckausgabe

7.4.1 Überblick

Verwendung

Zur Steuerung der Druckausgabe gab es lange Zeit individuelle JES2- bzw. JES3-JCL-Erweiterungen (/*OUTPUT bzw. //*FORMAT). Diese sind nun vereinheitlicht in einer OUTPUT-Anweisung.
Die Parameter der OUTPUT-Anweisungen ergänzen SYSOUT-DD-Anweisungen und gelten

- entweder implizit (als Default) als Ergänzung zu *allen* SYSOUT-DD-Anweisungen im gesamten Job oder in bestimmten Steps, sofern diese SYSOUT-DD-Anweisungen keinen expliziten Bezug aufweisen
- oder explizit als Ergänzung zu einer oder mehreren bestimmten SYSOUT-DD-Anweisungen im gesamten Job oder in bestimmten Steps

Parameter, die in einer SYSOUT-DD-Anweisung angegeben sind (z.B. Ausgabeklasse, COPIES), haben absoluten Vorrang vor Werten aus anderen Quellen. Parameter, die nicht in dieser SYSOUT-DD-Anweisung angegeben sind, können aus OUTPUT-Anweisungen ergänzt werden.

Mehrfache explizite oder implizite Rückbezug nahmen auf unterschiedliche OUTPUT-Anweisungen führen zu mehrfachen Ausgabevorgängen.

Gültigkeit

OUTPUT-Anweisungen gelten auf Jobdauer [Job Level], wenn sie nach der JOB-Anweisung und vor der ersten EXEC-Anweisung geschrieben werden. OUTPUT-Anweisungen gelten auf Stepdauer [Step Level], wenn sie nach der zu diesem Step gehörenden EXEC-Anweisung geschrieben werden.

Namens-, Operations-, Parameterfeld

Die formalen Regeln sind weitgehend analog der JOB-Anweisung.

- Das Namensfeld enthält den Output-Namen. Der Output-Name ist frei wählbar.
- Der Operationscode muss OUTPUT lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt.
- Das Parameterfeld enthält ausschließlich Schlüsselwortparameter. Alle Parameter sind optional.

Der Systemprogrammierer kann Standardwerte (Default-Werte) vorgeben. Ein Standardwert wird dann wirksam, wenn ein bestimmter Parameter nicht angegeben wird.

7.4.2 Schlüsselwortparameter: DEFAULT

Verwendung

Der DEFAULT-Parameter gibt an, ob diese OUTPUT-Anweisung als „Default-Anweisung“ und damit deren sonstige Parameter als „Default-Parameter“ gelten.

```
//outname OUTPUT DEFAULT=[Y|YES],<weitere Parameter>
```

Diese OUTPUT-Anweisung wird als Default-OUTPUT-Anweisung definiert.

Regeln

Eine SYSOUT-DD-Anweisung kann explizite oder implizite Rückbezugnahmen aufweisen zu vorangehenden OUTPUT-Anweisungen im selben Job.
Bei SYSOUT-DD-Anweisungen, in denen kein OUTPUT-Parameter angegeben ist (implizite Rückbezugnahme),

- wird zuerst eine DEFAULT-OUTPUT-Anweisung in *diesem* Step [Step Level] gesucht
- wird danach, nur wenn keine solche Anweisung in diesem Step existiert, eine DEFAULT-OUTPUT-Anweisung im Job [Job Level] gesucht

Wird eine solche Anweisung gefunden, werden bisher nicht festgelegte Werte daraus ergänzt. Andernfalls werden keine weiteren Werte ergänzt.

Bei SYSOUT-DD-Anweisungen, in denen tatsächlich ein OUTPUT-Parameter angegeben ist (explizite Rückbezugnahme),

- wird zuerst, wenn die Rückbezugnahme in der Form *.outname geschrieben ist, eine gleichnamige OUTPUT-Anweisung in diesem Step [Step Level] gesucht.
- Danach wird, nur wenn keine solche Anweisung in diesem Step existiert, eine gleichnamige OUTPUT-Anweisung im Job [Job Level] gesucht.
- Alternativ wird, wenn die Rückbezugnahme in der Form *.stepname.outname geschrieben ist, eine gleichnamige OUTPUT-Anweisung im angegebenen Step [Step Level] gesucht.

Wird eine solche Anweisung gefunden, werden bisher nicht festgelegte Werte daraus ergänzt. Andernfalls ergibt sich ein JCL-Fehler.

7.4.3 Schlüsselwortparameter: CLASS, FORMS, COPIES

Verwendung

Mit diesen Parametern können für Druckausgabe-Datenbestände angegeben werden:

- CLASS — eine Ausgabeklasse (Druckausgabeklasse)
- FORMS — die Bezeichnung des zu bedruckenden Formulars (max. 8-stelliger Name, installationsspezifisch)
- COPIES — die Anzahl der zu erzeugenden Kopien (wie im COPIES-Parameter der DD-Anweisung)

```
//outname   OUTPUT  CLASS=class,FORMS=formname,
//          COPIES=<wie COPIES in Systemausg.>
```

Es ist zu beachten, dass diese Werte nur dann wirksam werden, wenn sie in der DD-Anweisung, die auf diese OUTPUT-Anweisung explizit oder implizit Bezug nimmt, nicht schon definiert sind.

Achtung

Soll die CLASS-Angabe der OUTPUT-Anweisung wirksam werden, ist in der DD-Anweisung SYSOUT=(,) anzugeben.

7.4.4 Schlüsselwortparameter: DEST

Verwendung

Der DEST-Parameter gibt an, an welches Ziel [Destination] der Druckausgabe-Datenbestand weitergeleitet werden soll. Diese Angabe ist nur erforderlich, wenn der Bestand nicht über den bzw. die standardmäßig definierten Drucker ausgegeben werden soll.

```
//outname   OUTPUT DEST=ziel
```

Die Namen möglicher Ziele sind installationsabhängig und müssen dokumentiert sein.

7.4.5 Weitere Schlüsselwortparameter (Auswahl)

Verwendung

Auf den Trennblättern [Output Separator Pages] können Informationen für und über den Empfänger ausgedruckt werden. Dazu dienen die Parameter

- TITLE — Titel, Bezeichnung der Druckausgabe
- NAME — Name des Empfängers
- ADDRESS — Adresse des Empfängers
- DEPT — Abteilungsbezeichnung [Department] des Empfängers
- BUILDING — Gebäudebezeichnung des Empfängers
- ROOM — Raumbezeichnung des Empfängers

```
//outname    OUTPUT  TITLE=text,NAME=text,DEPT=text,
//                   BUILDING=text,ROOM=text
```

Jeder Text kann max. 60 Stellen umfassen. Bei der Verwendung von Sonderzeichen ist der entsprechende Text in Hochkommata zu setzen.

```
//outname    OUTPUT  ADDRESS=(text,text,text,text>
```

Jeder Text bei ADDRESS kann aus bis zu 4 Subparametern (gedruckt als 4 Zeilen) zu je max. 60 Stellen bestehen. Bei der Verwendung von Sonderzeichen ist der entsprechende Text in Hochkommata zu setzen.

7.5 Speicherverwaltung durch SMS

7.5.1 Überblick

Hintergrund

Das ständige Wachstum der Daten sowie die steigende Anzahl und Größe der Datenbestände führt dazu, dass die Verwaltung der Datenbestände und der Datenträger immer umfangreicher und komplexer wird. Hinzu kommt, dass der Anwender beim Anlegen eines Datenbestandes die Systemleistung, die Verfügbarkeit sowie die optimale Speicherplatz-Zuordnung nicht kennt (in der Regel auch nicht kennen kann) und deshalb nicht berücksichtigen kann.

Das Storage Management Subsystem (SMS) bietet eine umfangreiche Palette von Möglichkeiten, diese Aufgaben dem Anwender abzunehmen und wenigstens teilweise zu automatisieren.
In diesem Umfeld stehen eine Reihe von Programmen zur Verfügung, die unter dem Oberbegriff „Data Facility Storage Management Subsystem“ (DFSMS) zusammengefasst werden.

Beim Einsatz von SMS ergeben sich teilweise erhebliche Konsequenzen u.a. auf

- die JCL für Batch-Jobs
- die TSO-Kommandosprache für Dialoganwendungen
- das Dienstprogramm AMS.

Es hängt von der Installation ab, ob die Benutzung von SMS vorgeschrieben oder verboten ist, oder ob die konventionellen und die SMS-Methoden nebeneinander benutzt werden.

Schwerpunkte

Schwerpunktmäßig werden die Konsequenzen des Einsatzes von SMS für den Anwender erkennbar:

- Beim Anlegen eines Datenbestandes
  - Die Festlegung von UNIT und VOL=SER entfällt.
  - Der Bestand muss katalogisiert werden, d.h.
  - DISP=(NEW,CATLG[,DELETE]) ist zwingend.
- Bei der Automatisierung der Datensicherung.
- Bei der Automatisierung der Auslagerung wenig benutzter Datenbestände auf kostengünstige Datenträger.
- Bei der Automatisierung der Löschung unbenutzter Datenbestände.

Verwendung

Um die Möglichkeiten, die SMS bietet, auch nutzen zu können, müssen in jeder Installation umfangreiche Planungen und Vorbereitungen durchgeführt werden. Hierzu gehören u.a. folgende Maßnahmen:

- Definition von Datenklassen (**DATACLAS**). Eine Datenklasse [data class] enthält Definitionen für die logischen Eigenschaften von Datenbeständen wie Organisationsform, Satzlänge und Platzbedarf. Diese Eigenschaften können vom Anwender modifiziert bzw. ergänzt werden.
- Definition von Speicherklassen (**STORCLAS**). Eine Speicherklasse [storage class] enthält Definitionen für die physischen Eigenschaften von Datenträgern bezogen auf die Systemleistung, z. B. Zugriffsgeschwindigkeit, Einheiten mit Cache, mit Backup-Einrichtung (Dual Copy), usw.. Diese Eigenschaften können vom Anwender nicht modifiziert werden.
- Definition von Verwaltungsklassen (**MGMTCLAS**). Eine Verwaltungsklasse [management class] enthält Definitionen für organisatorische Maßnahmen, z. B. Backup-, Recovery- und Migration-Verfahren. Diese Eigenschaften können vom Anwender nicht modifiziert werden.
- Definition von Speichergruppen. In einer Speichergruppe [**storage group**] sind mehrere Datenträger mit gleichartigen Eigenschaften zusammengefasst. Zum Anlegen eines Datenbestandes wird eine solche Gruppe ausgewählt.
- Erstellen von **ACS-Routinen**. Diese Zuordnungs- und Auswahlroutinen [automatic class selection routines] verknüpfen die durch die drei Klassen definierten Einzeleigenschaften, prüfen die Berechtigung des Anwenders und wählen eine passende Speichergruppe für das tatsächliche Anlegen des Datenbestandes aus.

Zum Anlegen eines ‘SMS-managed Datenbestandes gibt der Anwender eine DATACLAS, eine STORCLAS und eine MGMTCLAS an. Durch den oben beschriebenen Mechanismus wählt das SMS selbst einen geeigneten Datenträger aus.
In jeder Installation sind Standardannahmen (Default-Klassen) für die drei Klassen festgelegt. Somit werden beim Weglassen einer oder aller entsprechenden Angaben die jeweilige Default-Klasse entsprechend dem LLQ (niedrigster Qualifier im DSN) ausgewählt.

Syntax beim Anlegen eines ‘SMS-managed‘ Datenbestandes

```
//ddname    DD    DSN=<Datenbestand>,DISP=(,CATLG,DELETE)
//                DATACLAS=<Datenklasse>,
//                STORCLAS=<Speicherklasse>,
//                MGMTCLAS=<Verwaltungsklasse>,
//                <sonst.Param.>
```

7.5.2 Datenklasse (DATACLAS)

Verwendung

Eine Datenklasse (DATACLAS) enthält eine vordefinierte Liste von Eigenschaften für die Datenbestands- und Speicherplatz-Zuordnungen sowohl für Nicht-VSAM- als auch für VSAM-Bestände. Allerdings ist es praktisch nicht sinnvoll, jede denkbare Kombination von Eigenschaften als eine eigene DATACLAS zu beschreiben. Es sind daher nur die am häufigsten benötigten Kombinationen als DATACLAS definiert. In der Regel wählt der Anwender die am besten passende DATACLAS aus und spezifiziert abweichende Eigenschaften mit Hilfe geeigneter Parameter in der DD-Anweisung.

DATACLAS=Datenklasse

Prinzipiell genügt beim Anlegen eines Datenbestandes die Angabe der ‘passenden‘ DATACLAS (neben DSN, DISP, STORCLAS, MGMTCLAS).
Die Eigenschaften einer DATACLAS können vom Anwender mit Hilfe zusätzlicher Parameter modifiziert bzw. ergänzt werden.

7.5.3 Schlüsselwortparameter: RECFM, RECORG, LRECL, u.ä.

Verwendung

Hiermit werden der Typ des anzulegenden Datenbestandes und einige weitere Eigenschaften festgelegt:

- RECFM Nicht-VSAM-Bestand
- RECORG VSAM-Bestand

RECFM=recfm

Das Satzformat [Record Format] kann sein: F, FB, V, VB, U, xxA, xxM. Die Einzelheiten wurde in einem früheren Abschnitt ausführlich beschrieben.

RECORG=[KSIESIRRILS]

Die Datensatz-Organisation [Record Organization] legt den Typ des VSAM-Bestandes fest, KSDS, ESDS, RRDS, LDS (wird hier nicht behandelt).

LRECL=rlen

Die Satzlänge [Logical Record Length] gibt an:

- die exakte Satzlänge (1...32760) bei RECFM=F/FB
- die maximale Satzlänge (1...32756) bei RECFM=V/VB

KEYLEN=klen,KEYOFF=koff

Die Länge des Schlüsselfeldes und sein Abstand [Offset] vom Satzanfang wird in der Regel nur bei VSAM-Beständen benutzt.

7.5.4 Schlüsselwortparameter: SPACE, AVGREC

Verwendung

Hiermit wird der Platzbedarf des anzulegenden Datenbestandes (auf Platte) festgelegt. Neben denjenigen Formen, die in einem früheren Abschnitt beschrieben wurden, existiert eine spezielle Form für ‘SMS-managed Datenbestände.

SPACE=(arlen,(primary[,secondary])[,RLSE]),AVGREC[U|K|M]

Mit dieser Form wird der Platzbedarf eines sequentiellen Datenbestandes oder eines VSAM-Datenbestandes festgelegt.

SPACE=(arlen,(primary[,secondary],dirblocks)[,RLSE]), AVGREC=[U|K|M]

Mit dieser Form wird der Platzbedarf eines untergliederten Datenbestandes festgelegt. Hierbei wirkt der SPACE-Parameter mit dem DSNTYPE-Parameter zusammen.

SPACE-Einheiten: arlen, primary, secondary

AVGREC-Einheiten

Hiermit wird festgelegt:

- arlen — die durchschnittliche Satzlänge [average record length] des anzulegenden Datenbestandes
- primary — die anfängliche Anzahl von Sätzen (Primärmenge) im anzulegenden Datenbestand
- secondary — die optionale Erweiterungsmenge in Sätzen (Sekundärmenge) im anzulegenden Datenbestand

Die Bedeutung der ‘primary‘ und ‘secondary‘ Werte wird festgelegt durch:

- AVGREC=U — diese Werte zählen einfach (Faktor 1)
- AVGREC=K — diese Werte zählen tausendfach (Faktor 1024)
- AVGREC=M — diese Werte zählen millionenfach (Faktor 1024*1024)

Die sonstigen Regeln für die Platzzuordnung wurden in einem früheren Abschnitt ausführlich beschrieben.

7.5.5 Schlüsselwortparameter: DSNTYPE

Verwendung

Hiermit wird beim Anlegen eines untergliederten Datenbestandes festgelegt, ob eine Bibliothek vom (konventionellen) Typ PDS [Partitioned Data Set] oder vom (modernen) Typ PDSE [Partitioned Data Set Extended] erstellt werden soll.
Für eine Bibliothek vom Typ PDS muss im SPACE-Parameter die Angabe dirblocks‘ vorhanden sein. Das Inhaltsverzeichnis [Directory] hat eine feste, unveränderliche Größe.
Für eine Bibliothek vom Typ PDSE wird keine Angabe ‘dirblocks‘ benötigt. Falls sie vorhanden ist, wird sie ignoriert. Das Inhaltsverzeichnis [Directory] hat hier keine feste Größe.

DSNTYPE=[PDS|LIBRARY]

Hiermit wird festgelegt:

- DSNTYPE=PDS Bibliothek vom Typ PDS
- DSNTYPE=LIBRARY Bibliothek vom Typ PDSE

7.5.6 Schlüsselwortparameter: LIKE, REFDD

Verwendung

Es besteht die Möglichkeit, für die Eigenschaften des anzulegenden Datenbestandes diejenigen eines anderen Bestandes als ‘Modell‘ zu nehmen.
Aus dem Modell können übernommen werden RECFM bzw. RECORG, LRECL, KEYLEN, KEYOFF, SPACE, AVGREC, DSNTYPE.
Eigenschaften, die durch entsprechende Parameter der aktuellen DD-Anweisung (neben LIKE bzw. REFDD) spezifiziert werden, haben Vorrang vor denjenigen des Modells.

LIKE=dsname

Als Modell dient der angegebene, existierende und katalogisierte Datenbestand auf Platte.

REFDD=*.ddname oder
REFDD=*.stepname.ddname oder
REFDD=*.stepname.procstepname.ddname

Als Modell dienen die Eigenschaften, die in derjenigen DD-Anweisung spezifiziert sind, auf die der Rückbezug verweist.

7.5.7 Speicherklasse (STORCLAS)

Verwendung

Eine Speicherklasse (STORCLAS) beschreibt Eigenschaften von Datenträgern und Geräten zur Datenspeicherung. Hierzu gehören u.a. folgende Eigenschaften:

- das Antwortzeitverhalten für sequentiellen Zugriff in Millisekunden
- das Antwortzeitverhalten für wahlfreien Zugriff in Millisekunden
- die überwiegende Nutzung (nur Lesen oder auch Schreiben)
- die durchschnittliche Datenübertragungsrate in MB/Sekunden
- die fortwährende Verfügbarkeit eines Datenträgers (mit Hilfe einer hardwaremäßig realisierten Datenträger-Spiegelung).

STORCLAS~Speicherklasse

Beim Anlegen eines Datenbestandes erfolgt die Angabe der gewünschten STORCLAS (neben DSN, DISP, DATACLAS, MGMTCLAS).
Die Eigenschaften einer STORCLAS können vom Anwender nicht modifiziert oder ergänzt werden.
Ein Datenbestand gilt dann als ‘SMS-managed’, wenn er beim Anlegen eine STORCLAS erhält. Allerdings kann eine STORCLAS auch aufgrund einer Default-Definition wirksam werden, d.h. ohne in der DD-Anweisung zu erscheinen.

7.5.8 Verwaltungsklasse (MGMTCLAS)

Verwendung

Eine Verwaltungsklasse (MGMTCLAS) beschreibt Verwaltungsmaßnahmen für Datenbestände. Hierzu gehören u.a. folgende Maßnahmen:

- das automatisierte Auslagern [Migration] und Zurückholen [Recall] von Datenbeständen
- die automatisierte Datensicherung [Backup] von Datenbeständen
- das Zurückkopieren [Restore] von Datenbeständen
- das automatisierte Löschen [Delete] von Datenbeständen, z.B. basierend auf Verfallsfristen
- das automatisierte Freigeben [Release] von unbenutztem Plattenplatz

Migration und Recall

Bei der Migration werden diejenigen Datenbestände, die während eines bestimmten Zeitraums nicht benutzt werden, automatisch auf einen anderen, kostengünstigen Datenträger übertragen. Bei Bedarf (Recall) werden sie auf einen geeigneten, ‘normalen‘ Datenträger zurückkopiert.

Backup und Restore

Beim Backup werden als Sicherungsmaßnahme Datenbestände in regelmäßigen zeitlichen Abständen auf einen anderen Datenträger (in der Regel Band) umkopiert. Im Bedarfsfall (Restore) können die Kopien als Ersatz für (beschädigte oder zerstörte) Originale zurückkopiert werden.

MGMTCLAS=Verwaltungsklasse

Beim Anlegen eines Datenbestandes erfolgt die Angabe der gewünschten MGMTCLAS (neben DSN, DISP, DATACLAS, STORCLAS).
Die Eigenschaften einer MGMTCLAS können vom Anwender nicht modifiziert oder ergänzt werden.

8. INCLUDE-Gruppen, JCL-Prozeduren

8.1 Allgemeines

8.1.1 Überblick

Gemeinsamkeiten

INCLUDE-Gruppen und JCL-Prozeduren sind Folgen von JCL-Anweisungen, die getrennt vom eigentlichen Job gespeichert werden. Sie erhalten Namen und werden unter diesem Namen als Elemente untergliederter Datenbestände abgespeichert. INCLUDE-Gruppen und JCL-Prozeduren können in beliebigen Jobs beliebig oft verwendet werden.

INCLUDE-Gruppen

Eine INCLUDE-Gruppe wird überwiegend dort verwendet, wo bestimmte, beliebige Sequenzen von JCL-Anweisungen häufig in (im Wesentlichen) unveränderter Form benutzt werden sollen.

JCL-Prozeduren

Eine JCL-Prozedur wird überwiegend dort verwendet, wo komplette Steps (d.h. EXEC-Anweisungen mit zugehörigen DD-Anweisungen) häufig in unveränderter oder veränderter Form benutzt werden sollen.

Veränderungsmöglichkeiten

JCL-Anweisungen können somit auftreten:

- normal‘ innerhalb von Jobs (d.h. außerhalb von INCLUDE-Gruppen und JCL-Prozeduren)
- innerhalb von INCLUDE-Gruppen
- innerhalb von JCL-Prozeduren.

JCL-Anweisungen können während der Jobausführung beeinflusst, abgeändert oder angepasst werden:

- in jedem Fall mit Hilfe von symbolischen Parametern (auch als Variablen bezeichnet)
- nur bei JCL-Prozeduren durch Ergänzen, Ersetzen oder Aufheben einzelner Parameter der EXEC- und/oder DD-Anweisungen
- nur bei JCL-Prozeduren durch Ergänzen, Ersetzen oder Aufheben kompletter DD-Anweisungen

8.2 JCLLIB-Anweisung

8.2.1 Überblick

Verwendung

Die INCLUDE-Gruppen und JCL-Prozeduren können als Elemente in einer oder mehreren standardmäßig festgelegten Prozedur-Bibliotheken (z.B. SYS1 .PROCLIB) liegen. Sie stehen damit allen Jobs zur Verfügung.
Aus Sicherheitsgründen haben die Anwender in der Regel keinen Schreibzugriff auf diese standardmäßigen Prozedur-Bibliotheken. Statt dessen kann jeder Anwender in einer oder mehreren beliebigen, privaten Bibliotheken (mit RECFM=FB,LRECL=80) selbst erstellte INCLUDE-Gruppen und JCL-Prozeduren abspeichern.
Sollen für die Ausführung eines Jobs private Bibliotheken benutzt werden, so sind sie dem JES2/3 durch die JCLLIB-Anweisung bekannt zu machen.

JCLLIB-Anweisungen gelten immer auf Jobdauer [Job Level]. Sie müssen

- nach der JOB-Anweisung und
- vor der ersten EXEC-Anweisung

geschrieben werden.

Namens-, Operations-, Parameterfeld

Die formalen Regeln sind weitgehend gleich wie bei der JOB-Anweisung.

- Das Namensfeld ist optional. Der angegebene Name ist frei wählbar.
- Der Operationscode muss JCLLIB lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt.
- Das Parameterfeld enthält nur den Schlüsselwortparameter ORDER.

8.2.2 Schlüsselwortparameter: ORDER

Verwendung

Der ORDER-Parameter gibt die privaten Bibliotheken an, in denen nach INCLUDE-Gruppen und/oder JCL-Prozeduren gesucht werden soll.

//[name] JCLLIB ORDER(bibliothek[,..])

Hierdurch wird dadurch die Reihenfolge für das Suchen nach den einzelnen Elementen (max. 15 katalogisierte Bibliotheken) festgelegt.

Regel

Ist keine JCLLIB-Anweisung angegeben oder ist die Suche in allen angegebenen Bibliotheken erfolglos, so wird in den standardmäßig festgelegten Prozedur-Bibliotheken (weiter-)gesucht.

8.3 INCLUDE-Anweisung

8.3.1 Überblick

Verwendung

Eine INCLUDE-Gruppe kann aus (nahezu) beliebigen und beliebig vielen JCL-Anweisungen bestehen. Folgende JCL-Anweisungen sind jedoch innerhalb einer INCLUDE-Gruppe nicht erlaubt:

- JOB-Anweisungen
- PROC-, PEND-Anweisungen
- JCLLIB-Anweisungen
- DD-Anweisungen für lnstream-Datenbestände (DD * bzw. DD DATA)
- JES2- bzw. JES3- spezifische JCL-Anweisungen.

Es können bis zu 15 INCLUDE-Anweisungen verschachtelt werden.

INCLUDE-Gruppen liegen als Elemente in einer oder mehreren standardmäßig festgelegten Prozedur-Bibliotheken (z.B. SYS1.PROCLIB) oder in privaten Bibliotheken, die mittels einer JCLLIB-Anweisung dem aktuellen Job zugeordnet sind.

Namens-, Operations-, Parameterfeld

Die formalen Regeln sind weitgehend gleich wie bei der JOB-Anweisung.

- Das Namensfeld ist optional. Der angegebene Name ist frei wählbar.
- Der Operationscode muss INCLUDE lauten. Er ist spaltenunabhangig, d h. die Beginnposition ist freigestellt.
- Das Parameterfeld enthält nur den Schlüsselwortparameter MEMBER.

8.3.2 Schlüsselwortparameter: MEMBER

Verwendung

Der MEMBER-Parameter gibt den Namen des einzufügenden Elementes, d.h. den Namen einer INCLUDE-Gruppe an. Das Element steht in einer privaten oder in einer standardmäßig festgelegten Prozedur-Bibliothek.

//[name] INCLUDE MEMBER=member

Die Inhalt der INCLUDE-Gruppe wird an derjenigen Stelle in den Job eingefügt, an der die INCLUDE-Anweisung steht.

8.4 Variable, SET-Anweisung

8.4.1 Überblick

Verwendung

Mit Hilfe von symbolischen Parametern (auch als Variablen bezeichnet) innerhalb von JCL-Anweisungen ist es möglich,

- einen Job
- eine INCLUDE-Gruppe
- eine JCL-Prozedur

möglichst allgemein zu formulieren, d.h. bestimmte Parameter und/oder Werte bei der Joberstellung *nicht* festzulegen. Die Variablen dienen als „Platzhalter'. Eine bestimmte Variable darf auch mehrfach verwendet werden. Die aktuellen Parameter bzw. Werte werden erst zum Zeitpunkt der Jobausführung festgelegt.

Symbolische Parameter (Variable)

Der Name eines symbolischen Parameters (einer Variablen) darf maximal 8 Zeichen (Stellen) lang sein, denen & vorangestellt wird.

Beispiel: &VAR1

Das erste Zeichen muss alphabetisch (A...Z, Großbuchstaben!) oder eines der Sonderzeichen § # $ sein. Die weiteren Zeichen müssen alphanumerisch (A. . .Z, 0.. .9) oder eines der Sonderzeichen § # $ sein.

Als Namen für Variable dürfen nicht verwendet werden:

- Namen, die als Schlüsselwort in EXEC-Anweisungen auftreten können (ACCT, ADDRSPC, COND, DPRTY, DYNAMNBR, PARM, PERFORM, PGM, PROC, RD, REGION, TIME)
- Namen, die mit SYS beginnen. Diese sind für Systemvariablen reserviert.

Bei der Erstellung des Jobs, der INCLUDE-Gruppe oder der JCL-Prozedur wird der Name einer Variablen dort eingesetzt, wo später (bei der Jobausführung) ein aktueller Wert eingesetzt werden soll.

Der aktuelle Wert für eine Variable wird festgelegt:

- mittels SET-Anweisung (bei Jobs, INCLUDE-Gruppen, JCL-Prozeduren) an beliebiger Stelle vor der erstmaligen Verwendung der Variablen
- als Standardwert (Default-Wert) bei der Prozedur-Definition (nur bei JCL-Prozeduren)
- als Wert beim Prozedur-Aufruf (nur bei JCL-Prozeduren).

8.4.2 Verwendung von Variablen

Regeln

Variable symbolisieren immer eine Zeichenkette. Die Zeichenkette darf maximal 255 Stellen lang sein und kann einen kompletten Parameter, einen Subparameter, Teile von Parametern oder Subparametern, aber auch Schlüsselwörter oder Teile davon umfassen.
Variablen dürfen nur im Parameterfeld von JCL-Anweisungen verwendet werden.
Variablen können auftreten:

- einzelnstehend (z.B. DSN&VAR1 ,DISP=OLD)
- kombiniert mit vorangehenden, konstanten Zeichenketten
  z.B. DSN=FILIALE.&VAR2 oder
  DSN=FILIALE.UMSATZ.JAHR&VAR3
- kombiniert mit nachfolgenden, konstanten Zeichenketten
  z.B. DSN=FILIALE.UMSATZ.&VAR4.97 oder
  DSN=&VAR5..UMSATZ

  Hinweis
  Zur Abgrenzung der Variablen von der nachfolgenden Zeichenkette ist die Variable mit einem Punkt abzuschließen. Falls ein Punkt folgen soll, sind also zwei Punkte anzugeben.
- mehrfach in Folge
  z.B. DSN=FILIALE.UMSATZ.&VAR6&VAR7 oder
  DSN=FLIALE.UMSATZ.&VAR6.&VAR7

  Hinweis:
  Zur Abgrenzung der vorangehenden Variablen von der nachfolgenden Variablen ist ein Punkt zulässig, aber nicht notwendig.

Der Wert einer Variablen

- kann als einfache Zeichenkette zugewiesen werden
  z.B. VAR1=FILIALE
- muss in Apostrophe eingeschlossen werden, wenn der Wert auch Sonderzeichen oder Leerstellen enthält
  z.B. VAR1='FILIALE.HAMBURG'
- kann ersatzlos aufgehoben werden
  z.B. VAR1=

  Achtung:
  Bei der Zuweisung an eine Variable wird das führende Zeichen & beim Namen der Variablen nicht angegeben.

Systemvariable

Systemvariable sind immer verfügbar. Die Zuweisung eines Wertes geschieht durch das System. &SYSUID enthält in der Regel die TSO-Userid des Benutzers, der den Job abgeschickt hat bzw. der Eigentümer des Jobs ist.

8.4.3 SET-Anweisung

Verwendung

Die SET-Anweisung dient zur Zuweisung einer Zeichenkette an Variable

- bei Jobs
- für INCLUDE-Gruppen
- für JCL-Prozeduren.

Namens-, Operations-, Parameterfeld

Die formalen Regeln sind weitgehend gleich wie bei der JOB-Anweisung.

- Das Namensfeld ist optional. Der angegebene Name ist frei wählbar.
- Der Operationscode muss SET lauten. Er ist spaltenunabhängig, d.h. die Beginnposition ist freigestellt.
- Das Parameterfeld enthält die Zuweisungen von Werten zu den Variablen.

//[name] SET variable=wert[,...]

Die Zuweisung erfolgt nach den im vorhergehenden Abschnitt erklärten Regeln.

Hinweise

Die SET-Anweisung kann an beliebigen Stellen (nach der JOB-Anweisung) benutzt werden.
Die SET-Anweisung wird immer an der Stelle im Job wirksam, an der sie geschrieben ist. Selbst wenn sie innerhalb einer IF/THEN-, ELSE-, ENDIF-Struktur erscheint, wird sie unabhängig von der Wahr-Falsch-Logik immer ausgeführt.

8.5 JCL-Prozeduren

8.5.1 Überblick

Verwendung

Eine JCL-Prozedur umfasst eine Folge von vordefinierten JCL-Anweisungen und besteht immer aus einem oder mehreren kompletten Steps. Unter Berücksichtigung dieses Grundsatzes kann eine JCL-Prozedur aus (nahezu) beliebigen und beliebig vielen JCL-Anweisungen bestehen. Folgende JCL-Anweisungen sind jedoch innerhalb einer JCL-Prozedur nicht erlaubt:

- JOB-Anweisungen
- JCLLIB-Anweisungen
- DD-Anweisungen zur Definition von Oatenbeständen auf Jobebene z.B. //JOBCAT DD, //JOBLIB DD
- DD-Anweisungen für lnstream-Daten bestände (DD * bzw. DD DATA)
- JES2- bzw. JES3- spezifische JCL-Anweisungen

Inhalt einer Prozedur (Prozedur-Definition)

PROC- und PEND-Anweisungen dienen zur Kennzeichnung des Anfangs bzw. des Endes einer Prozedur-Definition. Zwischen PROC- und PEND-Anweisung stehen JCL-Anweisungen, die zusammen einen oder mehrere komplette Steps bilden. Somit ist zumindest ein Step zu schreiben mit

- einer EXEC-Anweisung und der Angabe PGM=
- sowie (in der Regel) der/den zugehörigen DD-Anweisung(en).

Im Parameterfeld der JCL-Anweisungen innerhalb von Prozeduren ist die Verwendung von Variablen erlaubt. Prinzipiell können beliebige Zeichenketten durch Variable dargestellt werden.

Arten von Prozeduren

- Bibliotheks-Prozeduren (auch katalogisierte Prozeduren genannt) liegen entweder als Elemente in einer oder mehreren standardmäßig festgelegten Prozedur-Bibliotheken (z.B. SYS1.PROCLIB) oder in privaten Bibliotheken, die mittels einer JCLLIB-Anweisung dem aktuellen Job zugeordnet sind. Diese Prozeduren stehen prinzipiell allen Jobs zur Benutzung offen.
- lnstream-Prozeduren sind innerhalb eines Jobs definiert, nach der JOB-Anweisung und vor der ersten EXEC-Anweisung. Diese Prozeduren sind nur in diesem Job benutzbar

.

8.5.2 Definition von Bibliotheks-Prozeduren

Verwendung

Bibliotheks-Prozeduren, auch katalogisierte Prozeduren genannt, stehen prinzipiell allen Jobs zur Benutzung offen. Sie liegen als Elemente in einer Prozedur-Bibliothek.

Namensgebung

Der Name einer Bibliotheks-Prozedur wird festgelegt durch den Namen des Elementes [Members] der Bibliothek.
Eine Bibliotheks-Prozedur kann durch eine PROC-Anweisung eingeleitet werden.
Ein Name im Namensfeld der PROC-Anweisung kann angegeben werden.
Der Name ist formell bedeutungslos, sollte jedoch, um Irritationen zu vermeiden, identisch sein mit dem Namen der Prozedur, d.h. dem Namen des Elementes in der Bibliothek.
In der PROC-Anweisung können Default-Werte für Variablen festgelegt werden, die innerhalb der Prozedur verwendet werden sollen.
Eine Bibliotheks-Prozedur kann durch eine PEND-Anweisung beendet werden.
Ein Name im Namensfeld der PEND-Anweisung kann angegeben werden. Er ist formell bedeutungslos.

8.5.3 Definition von Instream-Prozeduren

Verwendung

Instream-Prozeduren sind nur in dem Job benutzbar, in dem sie definiert sind.
Innerhalb eines Jobs sind max. 15 Instream-Prozeduren definierbar.

Namensgebung

Eine Instream-Prozedur muss durch eine PROC-Anweisung eingeleitet werden.
Der Name einer Instream-Prozedur wird festgelegt durch den Namen im Namensfeld der PROC-Anweisung.
In der PROC-Anweisung können Default-Werte für Variablen festgelegt werden, die innerhalb der Prozedur verwendet werden sollen.
Eine Instream-Prozedur muss durch eine PEND-Anweisung beendet werden.
Ein Name im Namensfeld der PEND-Anweisung kann angegeben werden. Er ist formell bedeutungslos.

8.5.4 Benutzung von Prozeduren

Verwendung

Prozeduren können beliebig oft innerhalb eines Jobs benutzt (aufgerufen) werden. Dabei können diese Prozeduren unverändert benutzt oder nach bestimmten Regeln modifiziert werden. Es können bis zu 15 Aufrufe von JCL-Prozeduren verschachtelt werden.

EXEC-Anweisung

Die EXEC-Anweisung definiert den Beginn eines Steps. In einer EXEC-Anweisung wird

- entweder ein Programm mit EXEC PGM=progname
- oder eine JCL-Prozedur mit EXEC PROC=procname
- oder eine JCL-Prozedur mit EXEC procname

aufgerufen.

Der Beginn eines Steps innerhalb einer Prozedur wird ebenfalls durch eine EXEC-Anweisung definiert. Die Steps innerhalb einer Prozedur werden als Prozedursteps bezeichnet

Namens-, Operations-, Parameterfeld

Das Namensfeld enthält den Stepnamen, sowohl beim Aufruf eines Programms als auch beim Aufruf einer Prozedur. Die sonstigen formalen Regeln der EXEC-Anweisung bleiben unverändert.
Im Parameterfeld können Aufruf-Parameter angegeben werden.

Syntax eines Prozedur-Aufrufes

```
//stepname     EXEC [PROC=]procname,
//                  <aufrufparameter>
//procstep.outname   OUTPUT  <modifiz. OUTPUT-Anw.>
//procstep.ddname    DD      <modifiz. DD-Anw.>
//procstep.outname   OUTPUT  <zusätzl. OUTPUT-Anw.>
//procstep.ddname    DD      <zusätzl. ]D]D-Anw.>
```

8.5.5 Aufruf-Parameter

Arten und Wirkung

Parameter beim Aufruf einer Prozedur können sein:

- Wertzuweisungen an symbolische Parameter (Variable).
- Wertzuweisungen an Schlüsselwortparameter der EXEC-Anweisung, gültig *entweder* für jeden Prozedurstep *oder* für einen bestimmten Prozedurstep innerhalb der Prozedur.

Wertzuweisungen bewirken folgende Anderungen innerhalb der Prozedur:

- **Ergänzung**: Es werden Werte zu Variablen bzw. zu Schlüsselwortparametern definiert, denen bisher kein Wert zugewiesen worden war.
- **Ersetzung**: Es werden Werte zu Variablen bzw. zu Schlüsselwortparametern definiert, deren bisheriger Wert überschrieben wird.
- **Aufhebung**: Es werden Werte zu Variablen bzw. zu Schlüsselwortparametern ersatzlos aufgehoben.

Regeln für Variable

- Bei Wertzuweisungen an Variable müssen Zeichenketten, die Sonderzeichen beinhalten, in Hochkommata eingeschlossen werden.
- Wertzuweisungen an Variable, die beim Aufruf einer Prozedur erfolgen, haben Vorrang vor möglicherweise in der PROC-Anweisung angegebenen Default-Werten für diese Variablen. Ist auch in der PROC-Anweisung kein Default-Wert für diese Variable angegeben, so wird ein mittels SET-Anweisung festgelegter Wert wirksam.
- Wird einer Variablen kein Wert zugewiesen, tritt ein JCL-Fehler auf. Jobs mit solchen Fehlern werden nicht zur Verarbeitung angenommen.

Regeln für Schlüsselwortparameter

Wertzuweisungen an Schlüsselwortparameter der EXEC-Anweisung müssen in folgender Reihenfolge geschrieben werden:

- Zuerst Werte, die für jeden Prozedurstep innerhalb der Prozedur gelten. Die entsprechenden Schlüsselwörter werden wie üblich geschrieben. Die gleichnamigen Schlüsselwortparameter werden bei allen Prozedursteps wirkungslos.
- Dann in der Reihenfolge der Prozedursteps diejenigen Werte, die für einen bestimmten Prozedurstep innerhalb der Prozedur gelten. Den entsprechenden Schlüsselwörtern wird, durch einen Punkt getrennt, der Name des betroffenen Prozedursteps angehängt.

Sonderregeln

- Der Parameter PGM= kann nicht verändert werden.
- Der Parameter TIME= ohne angehängten Namen des Prozedursteps gilt als Zeitlimit für die Prozedur insgesamt. Eventuell angegebene Zeitlimits bei den einzelnen Prozedursteps werden wirkungslos.
- Der Parameter PARM= ohne angehängten Namen des Prozedursteps gilt nur für den ersten Prozedurstep.

8.5.6 Prozedur-Aufruf mit OUTPUT- und DD-Anweisungen

Wirkung von OUTPUT- und DD-Anweisungen

Beim Aufruf einer Prozedur können OUTPUT- und/oder OD-Anweisungen angegeben werden mit folgender Wirkung:

- **Modifizierung**: Es wird eine innerhalb der Prozedur vorhandene OUTPUT- bzw. DD-Anweisung dadurch verändert, dass Positions- oder Schlüsselwortparameter ergänzt, ersetzt oder aufgehoben werden.
- **Hinzufügung**: Es wird eine innerhalb der Prozedur bisher nicht vorhandene OUTPUT- bzw. DD-Anweisung zusätzlich definiert.

Wertzuweisungen für Parameter der OUTPUT- bzw. DD-Anweisungen bewirken folgende Änderungen innerhalb der Prozedur:

- **Ergänzung**: Es werden Werte zu Schlüsselwortparametern definiert, denen bisher kein Wert zugewiesen worden war.
- **Ersetzung**: Es werden Werte zu Schlüsselwortparametern definiert deren bisheriger Wert überschrieben wird.
- **Aufhebung**: Es werden Werte zu Schlüsselwortparametern ersatzlos aufgehoben.
- **Widerspruch** (nur bei DD-Anweisungen). Es werden Werte zu Positions- oder Schlüsselwortparametern geschrieben, die einen Widerspruch zu vorhandenen Parametern darstellen. Dadurch wird die Art der JCL-Anweisung verändert. Solche Widersprüche ergeben sich bei Verwendung von

  DUMMY (Hinweis: DSN=NULLFILE wirkt wie DUMMY)
  * bzw. DATA
  SYSOUT=
  DSN= bzw. UNIT=

Regeln für OUTPUT- und DD-Anweisungen

Die Namen der OUTPUT- bzw. DD-Anweisungen wird, durch einen Punkt getrennt, der Name des betroffenen Prozedursteps vorangestellt.
OUTPUT- und OD-Anweisungen müssen in der Reihenfolge der Prozedursteps, und dabei wiederum in folgender Teil-Reihenfolge geschrieben werden:

- Zuerst in der Reihenfolge der OUTPUT- bzw. DD-Anweisun gen diejenigen Anweisungen, die eine Modifizierung (Ergänzung, Ersetzung, Aufhebung, Widerspruch) vorhandener Anweisungen bewirken.
- Danach diejenigen OUTPUT- bzw. OD-Anweisungen, die eine Hinzufügung bisher nicht vorhandener Anweisungen bewirken.

Hinweis

Ist dem Namen einer OUTPUT- bzw. DD-Anweisung nicht der Name eines Prozedursteps vorangestellt, dann ist die Zuordnung dieser Anweisung zu einem bestimmten Prozedurstep etwas kompliziert geregelt. Diese Situation sollte daher besser vermieden werden!

8.5.7 Spezielle Möglichkeiten bei DD-Anweisungen

Ersatz für DD * bzw. DD DATA

* bzw. DD DATA darf innerhalb einer Prozedur-Definition nicht auftreten.
Es sind folgende Ausweichmöglichkeiten denkbar:

- Weglassen der entsprechenden DD-Anweisung innerhalb der Prozedur-Definition, aber Angabe dieser DD-Anweisung, zusammen mit den gewünschten Instream-Daten, in Form einer Hinzufügung beim Prozedur-Aufruf.
- Formulieren der entsprechenden DD-Anweisung innerhalb der Prozedur-Definition als DUMMY. Angabe dieser DD-Anweisung, zusammen mit den gewünschten Instream-Daten, in Form einer Modifizierung (Widerspruch) beim Prozedur-Aufruf.
- Formulieren der entsprechenden DD-Anweisung innerhalb der Prozedur-Definition als Datenbestand (RECFM=FB,LRECL=80) auf Platte. Speichern der eigentlich als Instream-Daten konzipierten Datensätze in diesem Datenbestand.

Behandlung von Verkettungen

Soll eine Verkettung mehrerer Datenbestände beim Prozedur-Aufruf modifiziert werden, so ist die komplette Verkettung zu wiederholen. Allerdings sind die Parameterfelder derjenigen DD-Anweisungen der Verkettung, die unverändert bleiben sollen, leer zu lassen.
Lediglich die Parameterfelder derjenigen DD-Anweisungen der Verkettung, die verändert bleiben sollen, sind anzugeben.

Sonderfall: Bei einer Verlängerung einer schon vorhandenen Verkettung ist sinngemäß zu verfahren. Es ist zuerst die komplette bisherige Verkettung (mit leeren Parameterfeldern) anzugeben. Anschließend sind die gewünschten zusätzlichen DD-Anweisungen mit entsprechenden Eintragungen in den Parameterfeldern zu schreiben.
Sonderfall: Bei einer Verkürzung einer schon vorhandenen Verkettung ist zu beachten, dass eine DD-Anweisung mit dem DUMMY-Parameter die Verkettung faktisch beendet. Somit genügt es, die komplette bisherige Verkettung (mit leeren Parameterfeldern) soweit anzugeben, wie die bisherige Verkettung beibehalten werden soll. Anschließend ist eine zusätzliche DD-Anweisung mit dem DUMMY-Parameter zu schreiben.

Arbeiten mit NULLFILE

Wenn eine Prozedur so definiert werden soll, dass ein bestimmter Datenbestand auf Band oder Platte verarbeitet werden kann, der Name des Datenbestandes aber noch nicht bekannt ist, so kann als „Platzhalter“ zunächst OSN=NULLFILE verwendet werden.
Beim Prozedur-Aufruf kann dann dieser Wert durch einen ‘richtigen‘ DSN ersetzt werden. Findet dieses Ersetzen aus irgend einem Grunde nicht statt, wirkt DSN=NULLFILE wie eine DUMMY-Angabe.

8.5.8 Rückbezugnahmen

COND-Parameter in EXEC-Anweisungen

Bei Verwendung im Zusammenhang mit Prozeduren kann der COND-Parameter in EXEC-Anweisungen in folgenden Formen benutzt werden:

COND=(code,op,stepname)
COND=(code,op,stepname.procstepname)

Parameter in DD-Anweisungen

Rückbezugnahmen bei DD-Anweisungen können verwendet werden bei:

- dem DSN-Parameter
- dem VOL-/VOLUME-Parameter in der Form VOL=REF=
- dem DCB-Parameter
- dem REFDD-Parameter (nur bei Einsatz von SMS)
- dem OUTPUT-Parameter

Bei Verwendung im Zusammenhang mit Prozeduren können diese Schlüsselwortparameter (abgekürzt ‘sw) in DD-Anweisungen in folgenden Formen benutzt werden:

sw=*.stepname.ddname
sw=*.stepname.procstepname.ddname

Angabe: stepname

Bei Verwendung dieser Angabe außerhalb von Prozeduren bezieht sich ‘stepname‘ auf einen vorangegangenen Step

```
//stepname   EXEC PGM=
```

außerhalb einer Prozedur.
Bei Verwendung dieser Angabe innerhalb einer Prozedur bezieht sich ‘stepname‘ auf einen vorangegangenen Prozedurstep

```
//procstepnm EXEC PGM=
```

innerhalb derselben Prozedur.

Angabe: stepname.procstepname

Bei Verwendung dieser Angabe bezieht sich ‘stepname‘ auf einen vorangegangenen Prozedur-Aufruf

```
//stepname   EXEC PROC=
```

und ‘procstepname‘ zusätzlich auf einen bestimmten Prozedurstep

```
//procstepnm EXEC PGM=
```

innerhalb dieser Prozedur.

8.5.9 Verschachtelte Prozeduren

Verwendung

Es können bis zu 15 Aufrufe von JCL-Prozeduren verschachtelt werden, indem innerhalb einer JCL-Prozedur eine andere JCL-Prozedur aufgerufen wird.

Zusätzliche Regeln für Variable

Wird einer Variablen beim Aufruf kein Wert zugewiesen, in der PROC-Anweisung kein Default-Wert zugeordnet und auch kein Wert mittels SET-Anweisung festgelegt, so wird der Wert der gleichnamigen Variablen aus der rufenden Prozedur wirksam.
Wird einer Variablen auf keine der genannten Arten ein Wert zugewiesen, tritt ein JCL-Fehler auf. Jobs mit solchen Fehlern werden nicht zur Verarbeitung angenommen.

Zusätzliche Regeln für Schlüsselwortparameter der EXEC-Anweisung sowie für OUTPUT- und DD-Anweisungen

Im Job können nur JCL-Anweisungen der „obersten“ JCL-Prozedur-Ebene modifiziert werden.
Eine JCL-Prozedur kann nur die JCL-Anweisungen der unmittelbar aufgerufenen JCL-Prozedur modifizieren, jedoch nicht diejenigen der indirekt aufgerufenen Prozeduren.

Hinweis

Prozedur-Definitionen können nicht verschachtelt werden, d.h. PROC- und PEND-Anweisungen innerhalb einer PROC-/PEND-Sequenz sind nicht erlaubt.

8.5.10 Jobablauf-Protokoll

MSGLEVEL-Parameter

Bei Angabe des MSGLEVEL-Parameters der JOB-Anweisung kann angegeben werden, ob JCL-Anweisungen, die aus Prozeduren stammen, im Jobablauf-Protokoll erscheinen.
Eine Angabe von MSGLEVEL=(1 ,x) bewirkt ein Erscheinen der aus Prozeduren stammenden JCL-Anweisungen im Jobablauf-Protokoll.
Eine Angabe von MSGLEVEL=(O,x) oder MSGLEVEL=(2,x) unterdrückt das Erscheinen der aus Prozeduren stammenden JCL-Anweisungen im Jobablauf-Protokoll.
Zumindest zur Analyse von Fehlern ist die Angabe MSGLEVEL=(1,x) daher meist sehr hilfreich.

Kennzeichnung im Jobablauf-Protokoll

Zur Kennzeichnung der Herkunft der einzelnen JCL-Anweisungen wird im Jobablauf-Protokoll folgende Form verwendet:
Markierung in Spalte 1:

X	JCL-Anweisung stammt aus Bibliotheks-Prozedur
+	JCL-Anweisung stammt aus lnstream-Prozedur

Markierung in Spalte 2:

X	unveränderte JCL-Anweisung aus Bibliotheks-Prozedur
+	unveränderte JCL-Anweisung aus lnstream-Prozedur
/	veränderte (modifizierte) JCL-Anweisung aus einer Prozedur

Markierung in Spalte 3:

*	Kommentar-Anweisung

8.5.11 Standardprozeduren

Verwendung

Zusammen mit MVS/ESA werden eine Reihe von „fertigen“ JCL-Prozeduren ausgeliefert. Diese werden auch als Standardprozeduren bezeichnet.
Viele dieser Standardprozeduren dienen zur Programmentwicklung, d.h. zur Realisierung der Arbeitsschritte Übersetzen-Binden-Ausführen in den wichtigsten Programmiersprachen.
Will man diese JCL-Prozeduren benutzen, so muss bekannt sein:

- der Name der zu benutzenden JCL-Prozedur
- der Inhalt (die Definition) der zu benutzenden JCL-Prozedur, wenn man sie beim Aufruf modifizieren möchte

JCL-Prozeduren für Übersetzen-Binden-Ausführen

Die Arbeitsschritte Übersetzen-Binden-Ausführen werden auch als „Compile-Link-Go“ (CLG) bezeichnet. Entsprechende JCL-Prozeduren haben daher einen Namen, der mit ...CLG endet.
Da in jeder Installation andere Konventionen für den detaillierten Ablauf dieses Verfahrens existieren, sind die entsprechenden JCL-Prozeduren oft nicht im „MVS/ESA-Originalzustand“, sondern sind meist installationsspezifisch angepasst.

Beispiel Standardprozedur COB2UCLG

Die Standardprozedur COB2UCLG dient der Übersetzung, dem Binden und der Ausführung von COBOL-Programmen.

- Im Step COB2 wird ein Quellmodul übersetzt und ein Objektmodul (temporär) erstellt.
- Im Step LKED wird der Bindevorgang ausgeführt und ein Lademodul (temporär) erstellt.
- Im Step GO wird das Programm (der Lademodul) ausgeführt.

Da diese Vorgänge in vielen Varianten ausgeführt werden können, ist häufig eine Modifizierung dieser Standardprozedur erforderlich.

Einige Modifizierungsmöglichkeiten

Die Arbeitsweise des Übersetzer-Programms [Compilers] im Step COB2 kann durch PARM-Angaben beeinflusst werden.
Fehler, die beim Übersetzungsvorgang erkannt werden, können separat in einer Datei mit dem DD-Namen SYSTERM als Druckausgabe ausgegeben werden.
Schließlich muss das zu übersetzende Programm in einer Datei mit dem DD-Namen SYSIN dem Übersetzer-Programm „geliefert“ werden.
Bei der Ausführung des Programms im Step GO sind, abhängig vom jeweiligen Programm, meist weitere DD-Anweisungen erforderlich. Bei diesem Programm ist der Programm-Dokumentation zu entnehmen, dass zwei Dateien mit den DD-Namen EING und AUSG benötigt werden.

Hinweis

Ohne ausreichende Programm-Dokumentation sind weder Jobs richtig erstellbar noch JCL-Prozeduren sinnvoll benutzbar und/oder modifizierbar.

weiter gehende Informationen

Details zu Standardprozeduren für Compile und Link siehe Schulungsunterlagen zu dem Thema z/OS Dienstprogramme.