Deutsch

FUJITSU Software

BS2000 OSD/BC V11.0
Einführung in das DVS

Benutzerhandbuch

Ausgabe Juni 2020

Kritik… Anregungen… Korrekturen…

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

Sie können uns Ihre Kommentare per E-Mail an bs2000services@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001: 2015 erfüllt.

Copyright und Handelsmarken

Inhaltsverzeichnis

1 Einleitung

Das vorliegende Handbuch beschreibt die Funktionen des Datenverwaltungssystems des BS2000. Das Datenverwaltungssystem (DVS) ist ein selbstständiges Teilsystem im BS2000. Es ist das Bindeglied zwischen den vom Benutzer gesteuerten Zugriffen auf Datenobjekte und den zentralen Gerätetreibern des Basissystems. Auch das DVS selbst nutzt Dienste des Basissystems.

Das DVS unterstützt in besonderem Maße die Datenverarbeitung auf gemeinschaftlichen Datenträgern, den „Public Volume Sets“ (kurz: „Pubsets“).

In einem Pubset werden einzelne Platten-Volumes zusammengefasst. Neben dem Home-Pubset, das alle für den Systemlauf notwendigen Dateien aufnimmt, können zusätzlich noch weitere Pubsets in das System aufgenommen (importiert) werden. Der Systemverwalter kann Benutzerkennungen und damit die Daten der Benutzer auf die einzelnen Pubsets so verteilen, dass eine für den Systembetrieb günstige Aufteilung erreicht wird. Performance und Ausfallsicherheit sind hierbei die entscheidenden Kriterien.

Für Sie als Benutzer des BS2000 bedeutet das: Solange Sie nicht explizit private Datenträger anfordern, werden Ihre Dateien auf dem Pubset angelegt, der Ihnen von der Systembetreuung als Standard-Pubset zugewiesen wurde. Sie müssen weder Datenträger noch Geräte anfordern. Auch die Speicherplatzverwaltung übernimmt das DVS.

In jedem Pubset wird für jeden Benutzer, der auf diesen Pubset zugreifen darf, ein Eintrag im Benutzerkatalog SYSSRPM geführt. Die Dateien der Benutzer werden im Dateikatalog TSOSCAT des Pubsets verwaltet. Durch Benutzerkennung und Katalogkennung – das ist die Kennung des Pubsets, auf dem der Katalog geführt wird – sind alle Benutzerdateien eindeutig gekennzeichnet. Gleichzeitig ist gewährleistet, dass nur dem Dateieigentümer Dateizugriffe gestattet sind – sofern er nicht explizit die Datei für „Fremdzugriffe“ freigibt.

Neben der Ablage von Dateien auf Pubsets unterstützt das DVS den Benutzer bei der Verwaltung von Daten auf privaten Datenträgern (z.B. Bandmedien) sowie auf Net-Storage. Net-Storage ist Speicherplatz, der über NFS an das BS2000 angeschlossen wird. In allen Fällen müssen die zu bearbeitenden Dateien im Dateikatalog eines Pubsets katalogisiert sein.

Funktionen des Datenverwaltungssystems

Das DVS ermöglicht Ihnen die Verarbeitung Ihrer Daten durch das Führen von Dateien und die verschiedenen Funktionen, die für die Dateiverarbeitung nötig sind. Die DVS-Funktionen lassen sich grob einteilen in:

- Erzeugen und Verwalten von Dateien inkl. Speicherplatzverwaltung
- Verwalten von Katalogen
- Bereitstellen von Dateien und Dateibearbeitung über die Zugriffsmethoden
- Zuordnen von Dateien zu Programmen

Darüber hinaus bietet Ihnen das DVS die Möglichkeit, Datenschutz und Dateischutz-Merkmale auf Dateiebene zu definieren. Die Datensicherheit wird vom DVS z.B. beim Dateizugriff durch das Setzen von Sperren unterstützt.

Die Funktionen des DVS sind realisiert über Assemblerschnittstellen, die im Handbuch „DVS-Makros“ [1] beschrieben sind, und Kommandoschnittstellen, die im Handbuch „Kommandos“ [3] beschrieben sind.

Versionsliste relevanter Software-Produkte

Die im Handbuch aufgeführten Software-Produkte beziehen sich auf folgende Versionen:

- DAB V9.5

- HSMS V12.0
- HIPLEX-MSCF V11.0
- SECOS V5.5

1.1 Zielsetzung und Zielgruppen des Handbuchs

Das Handbuch wendet sich an Benutzer, die mithilfe der Assembler-Programmierschnittstellen oder mithilfe der DVS-Kommandos Dateien verarbeiten oder verwalten, die Datenschutz, Dateischutz, Datensicherheit usw. über Makros oder Kommandos steuern wollen und die eine allgemeine Einführung in die Funktionen des DVS benötigen.

Das Handbuch enthält eine allgemeine Beschreibung der Funktionsweise des DVS und dient zur Einarbeitung in das DVS und der Darstellung von Zusammenhängen. Die DVS-Schnittstellen sind ausführlich in den Handbüchern „DVS-Makros“ [1] und „Kommandos“ [3] beschrieben. Die im Handbuch genannten Kommandos und Makros, die nicht mit einem Literaturhinweis versehen sind, können dort nachgelesen werden.
Andere Literaturhinweise sind im Text in der Regel als Kurztitel angegeben. Das Literaturverzeichnis enthält die vollständigen Titel.

1.2 Konzept des Handbuchs

Im Einzelnen enthält das Handbuch folgende Themenschwerpunkte:

Übersicht über die Schnittstellen der DVS-Funktionen

Hier werden die grundsätzlichen Funktionen des DVS, sowie die jeweils dafür zur Verfügung stehenden Makros und Kommandos in tabellarischer Form vorgestellt.

Datenträger und Dateien im BS2000, Dateiverwaltung und Dateischutz

Es wird ein Überblick über die im BS2000 verfügbaren Datenträger gegeben.
Es werden die verschiedenen Dateitypen/Dateistrukturen definiert und beschrieben sowie Datei- und Katalogverwaltung, Dateizugriff, Datei- und Datenschutz.

Dateien auf Platte und Band

Es werden die verschiedenen Datenträger, die Datenträger- und Geräteverwaltung, Kennsätze für Banddateien sowie Kennsatzverarbeitung, Speicherverwaltung sowie MF/MV-Sets beschrieben.

Öffnen und Schließen von Dateien

Dieser Teil enthält die Beschreibung der DVS-internen OPEN- und CLOSE-Verarbeitung.

Zugriffsmethoden

In diesem Teil werden allgemein die Satz-, Block- und Plattenformate beschrieben sowie die Zugriffsmethoden UPAM, FASTPAM, DIV, SAM, BTAM, EAM und ISAM.

Anhang

Der Anhang enthält Tabellen und Listen, auf die an mehreren Stellen des Handbuchs Bezug genommen wird. Die Geräte- und Volumetyp-Tabelle finden Sie im Handbuch „Systeminstallation“ [15]. Informationen zu den Volumetypen bei der Bandverarbeitung sowie die Bedeutung der Ausgabewerte von SHOW-Kommandos der Geräteverwaltung finden Sie im Handbuch „Kommandos“ [3].

Readme-Datei

Funktionelle Änderungen der aktuellen Produktversion und Nachträge zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter *http://bs2manuals.ts.fujitsu.com* zur Verfügung. Alternativ finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen unter BS2000

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie im BS2000-System die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen.
Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

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

1.3 Änderungen gegenüber dem Vorgänger-Handbuch

Das Handbuch „DVS-Einführung“ wurde zuletzt zur BS2000 OSD/BC V10.0 aufgelegt. Folgende wesentliche Änderungen haben sich seit der letzten Ausgabe ergeben:

- Im Abschnitt „Net-Storage“ im Kapitel „Datenträger“ wird die Verarbeitung von SAM-Node-Files beschrieben, die ab BS2000 OSD/BC V11.0 unterstützt werden.
- Für Node-Files beschreibt das Dateiattribut NET-CODED-CHARACTER-SET (NETC-CS), wie die tatsächliche Codierung auf dem Net-Storage bei der Verarbeitung bzw. Darstellung im BS2000 umzusetzen ist. Das Dateiattribut wird für Node-Files unterstützt, ausgewertet wird der Eintrag aber nur für SAM-Node-Files. Für PAM-Node-Files, die in BS2000 OSD/BC V10.0 angelegt wurden, wird implizit der Wert *NO-CONV angenommen.
  Das NET-CODED-CHARACTER-SET kann explizit im Kommando CREATE-FILE bzw. MODIFY-FILE-ATTRIBUTES eingestellt werden. Voreingestellt ist, dass der NETCCSN einer neu anzulegenden Datei aus der benutzerspezifischen Einstellung ermittelt wird. Beim Einrichten des Benutzers wird für diese Voreinstellung wiederum der Systemparameter NETCODE zugrundegelegt.
- Mit IMPORT-NODE-FILE können nun auch Node-Files als SAM-Dateien importiert werden.
- Das Kommando SHOW-FILE-ATTRIBUTES gibt den Wert des Last Byte Pointers als LAST-BYTE im Informationsteil *Allocation* bzw. in S-Variablen aus, wenn das entsprechende Valid-Bit im Katalogeintrag der Datei gesetzt ist.
  Außerdem können Dateien nach den Eigenschaften des Last Byte Pointers selektiert werden.
- Der neue Systemparameter ISBLKCTL steuert beim Anlegen von ISAM-Dateien das Dateiattribut BLK-CONTR, falls der Anwender keine Angabe dazu macht. Die Voreinstellung ISBLKCTL=c'NONKEY' bewirkt, dass standardmäßig auch auf K-Platten NK-ISAM-Dateien (Dateiattribut BLK-CONTR=DATA) angelegt werden. Diese Voreinstellung berücksichtigt, dass K-ISAM in der Praxis immer mehr an Bedeutung verliert.
- Ab BS2000 OSD/BC V11.0B:
  Zur Unterstützung von Versions-Backup (ab HSMS V12.0) wurde der Dateikatalog um das Dateiattribut NUM-OF-BACKUP-VERS erweitert. Der dort definierte Wert (Wertebereich 0 bis 32) legt fest, ob die Datei am Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im Versions-Backup-Archiv gesichert werden sollen. Der Wert 0 bedeutet keine Teilnahme. Temporäre Dateien, Dateien auf privater Platte, Banddateien und Dateigenerationen werden vom Versions-Backup nicht unterstützt, d.h. bei diesen Dateien ist nur der Wert 0 möglich. Die Einstellung erfolgt im Makro CATAL (Operand NUM_OF_BACKUP_VERS) oder in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES (Operand NUM-OF-BACKUP-VERS). Ohne explizite Angabe wird bei Dateien, für die Versions-Backup unterstützt wird, der im Systemparameter NUMBACK definierte Wert übernommen.

Makro / Operand oder RC	Inhalt der Änderung
CATAL	Neuer Operand: NETCCS bestimmt das NET-CODED-CHAR-SET.
	Neuer Operand ab V11.0B: NUM_OF_BACKUP_VERS bestimmt, ob die Datei am Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im Versions-Backup-Archiv gesichert werden sollen.
FSTAT	Neuer Operand: LBPOINT zur Auswahl von Dateien abhängig vom Wert des Last Byte Pointers. Der Ausgabereich enthält auch das Dateiattribut NETCCS.

	Neuer Operand ab V11.0B: VERSION-BACKUP zur Auswahl von Dateien abhängig davon, ob die Datei am Versions-Backup teilnehmen kann. Die Teilnahme ist nur möglich, wenn für das Dateiattribut NUM-OF-BACKUP-VERS ein Wert > 0 definiert ist. Das Ausgabefeld #BACK-VERS enthält den Wert für das Dateiattribut NUM-OF-BACKUP-VERS.
IMPNFIL	Neuer Operandenwert: FILESTR=*SAM importiert Node-Files als SAM-Datei.

Die Makros sind in „DVS-Makros“ [1] vollständig beschrieben.

Übersicht der Änderungen an der DVS-Kommandoschnittstelle

Kommando / Operand	Inhalt der Änderung
CREATE-FILE	Neuer Operand: NET-CODED-CHAR-SET bestimmt das NET-CODED-CHAR-SET für Node-Files.
	Neuer Operand ab V11.0B: NUM-OF-BACKUP-VERS bestimmt, ob die Datei am Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im Versions-Backup-Archiv gesichert werden sollen.
IMPORT-NODE-FILE	Neuer Operandenwert: FILE-STRUCTURE=*SAM importiert Node-Files als SAM-Datei.
MODIFY-FILE-ATTRIBUTES	Neuer Operand: NET-CODED-CHAR-SET bestimmt das NET-CODED-CHAR-SET für Node-Files
	Neuer Operand ab V11.0B: NUM-OF-BACKUP-VERS bestimmt, ob die Datei am Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im Versions-Backup-Archiv gesichert werden sollen.
SHOW-FILE-ATTRIBUTES	Neuer Operand: LAST-BYTE-POINTER selektiert Dateien nach dem Wert des Last Byte Pointers. Ausgabe nach SYSOUT und in S-Variablen erweitert für Last Byte Pointer und NETCCS.
	Neuer Operand ab V11.0B: VERSION-BACKUP selektiert Dateien abhängig davon, ob sie am Versions-Backup teilnehmen können. Die Teilnahme ist nur möglich, wenn für das Dateiattribut NUM-OF-BACKUP-VERS ein Wert > 0 definiert ist. Bei Ausgabe nach SYSOUT wird das Dateiattribut NUM-OF-BACKUP-VERS im Ausgabefeld #BACK-VERS angezeigt, bei Ausgabe in S-Variablen in `var(*list).NUM-OF-BACKUP-VERS`.

Die Kommandos sind im Handbuch „Kommandos“ [3] vollständig beschrieben.

Allgemeine Änderung

Die Unterstützung des GS (Global Storage) als nicht-flüchtiges Cache-Medium entfällt.

Ab BS2000 OSD/BC V11.0 wird der Anschluss von Peripherie nur noch über Typ FC-Kanal unterstützt. Mit Entfallen des Typ S-Kanals entfällt auch die Unterstützung der Funktion „Mehrrechner-benutzbare Privatplatte“ (SPD).

Verschlüsseln und Entschlüsseln von Dateien erfolgt ab BS2000 OSD/BC V11.0 über das Subsystem CRYPT, siehe Handbuch „CRYPT“ [24].

1.4 Darstellungsmittel

Die Metasyntax der in diesem Handbuch beschriebenen Anweisungen finden Sie im Handbuch „Kommandos“ [3]. Operanden, die sich nur an Anwender mit Privilegierung richten, sind in den Formaten grau unterlegt.

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

Dieses Zeichen kennzeichnet Hinweise auf wichtige Informationen

Dieses Zeichen kennzeichnet einen Warnhinweis, der auf die Möglichkeit des Datenverlustes oder anderer ernsthafter Schäden an Daten hinweist.

`[ ]` Literaturhinweise werden im Text in Kurztiteln mit einer Verweisnummer in eckigen Klammern angegeben. Der vollständige Titel jeder Druckschrift, auf die verwiesen wird, ist im Literaturverzeichnis ab "Literatur" aufgeführt.

`Eingabe` In Anwendungsbeispielen sind Eingaben an das System und Ausgaben des Systems in `Schreibmaschinenschrift` dargestellt.

2 Übersicht über die Schnittstellen der DVS- Funktionen

Die folgenden Tabellen bieten eine Übersicht über die Schnittstellen der DVS-Funktionen. In den einzelnen Kapiteln dieses Handbuchs wird häufig auf diese Schnittstellen Bezug genommen. Die Tabellen sind – sofern beide Schnittstellen angeboten werden – nach Kommando- und Programmschnittstelle (Makro) aufgeteilt. Zur Beschreibung der Schnittstellen mit allen Operanden siehe die Handbücher „Kommandos“ [3], „DVS-Makros“ [1] und „Makroaufrufe an den Ablaufteil“ [2].

Neben den DVS-Kommandos wurden auch einige NDM-Kommandos (NDM = Nucleus Device Management) aufgenommen, die den Benutzer bei Geräte- und Datenträgerverwaltung unterstützen, die Informationen über Benutzereinträge liefern usw.

Tabelle der DVS-Makros

Makro	Kurzbeschreibung
ADDPLNK	*/SAM:* Poolkettungsnamen definieren
BTAM	Steuert alle BTAM-Aktionen
CATAL	Katalogeintrag bearbeiten
CHKFAR	Zugriffsrechte auf Datei überprüfen
CHNGE	TFT-Eintrag ändern
CLOSE	Datei schließen
COMPFIL	Zwei Plattendateien vergleichen
COPFILE	Datei kopieren
CREAIX	*/SAM:* Sekundärschlüssel für ISAM-Datei erzeugen
CREPOOL	*/SAM:* ISAM-Pool erzeugen
DECFILE	Verschlüsselte Datei in unverschlüsselte Datei umwandeln
DELAIX	*/SAM:* Sekundärschlüssel einer ISAM-Datei löschen
DELPOOL	*/SAM:* ISAM-Pool löschen/freigeben
DIV	Dateizugriff über virtuellen Adressraum
DROPTFT	Gesperrte TFT-Einträge freigeben
EAM	Makroaufruf (Typ R)
ELIM	*/SAM:* Satz streichen
ENCFILE	Unverschlüsselte Datei in verschlüsselte Datei umwandeln
ERASE	Dateien löschen

EXLST	Exit-Adressenliste anlegen (Typ O)
EXRTN	Rücksprung aus Fehlerroutinen (Typ R)
FCB	Definieren Dateisteuerblock (Typ O)
FCBAD	FCB-Adressen anlegen (Typ O)
FEOV	*BTAM/SAM:* Band abschließen
FILE	Dateimerkmale definieren / Dateiverarbeitung steuern
FILELST	Variable Operandenbereiche für FILE-Makro erzeugen
FPAMACC	*FASTPAM:* Dateizugriffe formulieren
FPAMSRV	*FASTPAM:* Verwaltungsaufrufe formulieren
FSTAT	Kataloginformation anfordern
GET	*ISAM/SAM:* Nächsten Satz lesen
GETFL	*ISAM:* Lesen Satz nach Markierung
GETKY	*ISAM:* Satz lesen mit angegebenem Schlüssel
GETR	*ISAM:* Sequenziell „rückwärts“ Lesen
IDBPL	*BTAM:* BTAM-Operandenliste (Typ O)
IDFCB	Versorgung eines FCB mit symbolischen Namen (Typ O)
IDFCBE	Versorgung der FCBE mit symbolischen Namen (Typ O)
IDMCB	Versorgung des MFCB (EAM-Steuerblock mit symbolischen Namen
IDPPL	*UPAM:* PAM-Operandenliste
IMPNFIL	Katalogeintrag für Node-Files erstellen (importieren)
IMPORT	Katalogeintrag für Dateien erstellen (importieren)
INSRT	*ISAM:* Satz einfügen (Typ R)
ISREQ	*ISAM:* Sperre aufheben (Typ O)
LBRET	Rückkehr aus Benutzer-Kennsatzroutine (Typ R)
LFFSNAP	Dateien von einem Snapset auflisten
LJFSNAP	Jobvariablen von einem Snapset auflisten
MAILFIL	Datei oder Bibliothekselement per E-Mail an eine Benutzerkennung versenden
NDWERINF	*BTAM:* Status-Bytes abfragen

OPEN	Datei eröffnen (Typ R)
OSTAT	*/SAM:* Information über eröffnete Dateien (Typ R)
PAM	*UPAM:* UPAM-Aktionen ausführen
PUT	*/SAM/SAM:* Satz schreiben
PUTX	*/SAM/SAM:* Satz ersetzen
RDTFT	Informationen aus TFT und TST
RELTFT	TFT-Eintrag löschen
REMPLNK	*/SAM:* Poolkettungsname löschen
RETRY	*/SAM:* Makroaufruf wiederholen
RELSE	Block abschließen
RFFSNAP	Dateien von einem Snapset restaurieren
RJFSNAP	Jobvariablen von einem Snapset restaurieren
SETL	*/SAM/SAM:* Positionieren in der Datei
SHOPLNK	Informationen über ISAM-Pool-Kettungsnamen ausgeben
SHOPOOL	Informationen über ISAM-Pool ausgeben
SHOWAIX	Informationen über Sekundärschlüssel ausgeben
STORE	*/SAM:* Satz speichern
VERIF	Datei wiederherstellen

Tabelle 1: DVS-Makros

DVS-Makros und DVS-Kommandos mit gleicher Funktionalität

Makro	Kommando	Funktion
ADDPLNK	ADD-ISAM-POOL-LINK	Poolkettungsnamen für einen ISAM-Pool definieren
CATAL	CREATE-FILE	Katalogeintrag erstellen
	CREATE-FILE-GROUP	Dateigenerationsgruppen definieren
	MODIFY-FILE-ATTRIBUTES	Schutzmechanismen definieren
	MODIFY-FILE-GROUP-ATTRIBUTES	Schutzmechanismen definieren
CHNGE	CHANGE-FILE-LINK	Dateikettungsnamen in der TFT ändern

COMPFIL	COMPARE-DISK-FILES	Zwei Plattendateien vergleichen
COPFILE	COPY-FILE	Datei kopieren
CREAIX	CREATE-ALTERNATE-INDEX	Sekundärindex für ISAM-Datei erzeugen
CREPOOL	CREATE-ISAM-POOL	ISAM-Pool anlegen
DECFILE	DECRYPT-FILE	Verschlüsselte Datei in unverschlüsselte Datei umwandeln
DELAIX	DELETE-ALTERNATE-INDEX	Sekundärindizes einer ISAM-Datei löschen
DELPOOL	DELETE-ISAM-POOL	ISAM-Pool löschen/freigeben
DROPTFT	UNLOCK-FILE-LINK	Gesperrte TFT-Einträge freigeben
ENCFILE	ENCRYPT-FILE	Unverschlüsselte Datei in verschlüsselte Datei umwandeln
ERASE	DELETE-FILE	Datei(en) löschen
	DELETE-FILE-GENERATION	Dateigenerationen löschen
	DELETE-FILE-GROUP	Dateigenerationsgruppe(n) löschen
	DELETE-SYSTEM-FILE	Systemdatei(en) löschen
	EXPORT-FILE	Datei(en) exportieren
	EXPORT-NODE-FILE	Node-File(s) exportieren
FILE	CREATE-FILE	Datei erzeugen
	CREATE-FILE-GENERATION	Dateigeneration erzeugen
	CREATE-TAPE-SET	TST-Eintrag erstellen
	EXTEND-TAPE-SET	TST-Eintrag erweitern
	IMPORT-FILE	Datei importieren
	MODIFY-FILE-ATTRIBUTES	Merkmale einer Datei ändern
	MODIFY-FILE-GENERATION-SUPPORT	Merkmale einer Dateigeneration ändern
	ADD-FILE-LINK	TFT-Eintrag erstellen
FSTAT	IMPORT-FILE	Informationen aus dem Dateikatalog zur Verfügung stellen

	SHOW-FILE-ATTRIBUTES	Dateimerkmale ausgeben
IMPNFIL	IMPORT-NODE-FILE	Katalogeintrag für Node-Files erstellen (importieren)
IMPORT	CHECK-IMPORT-DISK-FILE	Importieren von Dateien vorab prüfen
	IMPORT-FILE	Katalogeintrag für Dateien erstellen (importieren)
LFFSNAP	LIST-FILE-FROM-SNAPSET	Dateien von einem Snapset auflisten
LJFSNAP	LIST-JV-FROM-SNAPSET	Jobvariablen von einem Snapset auflisten
MAILFIL	MAIL-FILE	Datei oder Bibliothekselement per E-Mail an eine Benutzerkennung versenden
RDTFT	SHOW-FILE-LINK	Informationen aus der TFT zur Verfügung stellen
RELTFT	REMOVE-FILE-LINK	TFT-Eintrag aufheben
	DELETE-TAPE-SET	TST-Eintrag löschen
REMPLNK	REMOVE-ISAM-POOL-LINK	Poolkettungsnamen löschen
RFFSNAP	RESTORE-FILE-FROM-SNAP-SET	Dateien von einem Snapset restaurieren
RJFSNAP	RESTORE-JV-FROM-SNAP-SET	Jobvariablen von einem Snapset restaurieren
SHOWAIX	SHOW-INDEX-ATTRIBUTES	Informationen über Sekundärschlüssel einer ISAM-Datei ausgeben
SHOPLNK	SHOW-ISAM-POOL-LINK	Zuordnung von ISAM-Pools zu ISAM-Pool-Kettungsnamen ausgeben
SHOPOOL	SHOW-ISAM-POOL-ATTRIBUTES	Informationen über einen ISAM-Pool ausgeben
VERIF	CHECK-FILE-CONSISTENCY	Dateikonsistenz wiederherstellen
	REMOVE-FILE-ALLOCATION-LOCKS	Verfügbarmachen von Dateien, Dateigenerationen und FFGs, die wegen System- oder Auftragsabbruch gesperrt sind
	REPAIR-DISK-FILES	Wiederherstellen von Dateien, Dateigenerationen und FFGs, die wegen System- oder Auftragsabbruch gesperrt sind

Tabelle 2: Gegenüberstellung DVS-Makros / DVS-Kommandos

DVS-Kommandos ohne entsprechenden Makroaufruf

Kommando	Funktion
ADD-CRYPTO-PASS-WORD	hinterlegt das Crypto-Kennwort zur Entschlüsselung von verschlüsselten Dateiinhalten in der Kennworttabelle der Task
ADD-PASSWORD	Kennwort in die Kennworttabelle des Auftrags eintragen
CONCATENATE-DISK-FILES	SAM-Dateien verketten
EDIT-FILE-ATTRIBUTES EDIT-FILE-GROUP-ATTRIBUTES EDIT-FILE-GENERATION-SUPPORT	aktiviert den geführten Dialog des entsprechenden MODIFY-Kommandos und ermöglicht das „Editieren“ eines bestehenden Katalogeintrags
EDIT-FILE-LINK	aktiviert den geführten Dialog des ADD-FILE-LINK-Kommandos und ermöglicht das „Editieren“ eines bestehenden TFT-Eintrags
LIST-NODE-FILES	listet Node-Files eines Net-Storage-Volumes auf
LOCK-FILE-LINK	TFT-Eintrag sperren, sodass er erst nach einem UNLOCK-FILE-LINK-Kommando freigegeben wird.
REMOVE-CRYPTO-PASSWORD	entfernt das Crypto-Kennwort aus der Kennworttabelle der laufenden Task
REMOVE-PASSWORD	Kennwort aus der Kennworttabelle des Auftrags löschen
REPAIR-FILE-LOCKS	unberechtigt sitzende Dateisperren beseitigen
RESTART-PROGRAM	Programm an einem Punkt starten, der mit WRCPT- (oder CHKPT-) Makroaufruf gesichert wurde.
SHOW-BLOCK-TO-FILE-ASSIGNMENT	privilegiertes Kommando: Anzeigen von Dateien, in denen die angeforderten Blöcke liegen
SHOW-FILE	Ausgabe einer Datei auf SYSOUT
SHOW-FILE-LOCKS	Sperren einer Datei anzeigen
START-FILE-CACHING	Caching von bereits geöffneten Dateien starten
STOP-FILE-CACHING	Caching einzelner Dateien beenden

Tabelle 3: DVS-Kommandos ohne entsprechenden Makroaufruf

2.1 Wartung des Dateibestandes

Zur Wartung des Dateibestandes gehören nicht nur Erstellen, Kopieren oder Löschen von Dateien, sondern auch die Pflege des Dateikatalogs durch den Benutzer. Informationen zum Thema „Restaurieren“ finden Sie unter Abschnitt „Datensicherheit“.

Makros

Makro	Operanden	Kurzbeschreibung
CATAL		erstellt oder ändert Katalogeinträge
COMPFIL		vergleicht zwei Plattendateien
COPFILE		kopiert Dateien
ERASE		löscht Dateien / exportiert Dateien
FILE		erstellt einen Katalogeintrag und reserviert Speicherplatz für nicht katalogisierte Dateien
	SPACE	reserviert Speicherplatz oder gibt ihn frei
	NFTYPE	erstellt einen Katalogeintrag für den Dateityp BS2000-Datei oder Node-File auf Net-Storage
	STATE	erstellt einen Katalogeintrag für Dateien auf privaten Datenträgern
FSTAT		gibt Informationen aus dem Dateikatalog aus
IMPNFIL		erstellt Katalogeintrag für Node-Files (importieren)
IMPORT		erstellt Katalogeinträge für Dateien (importieren)
MAILFIL		sendet Datei oder Bibliothekselement per E-Mail an eine Benutzerkennung
VERIF		rekonstruiert beschädigte Dateien

Kommandos

Kommando	Kurzbeschreibung
CREATE-FILE CREATE-FILE-GROUP CREATE-FILE-GENERATION	erstellt einen Katalogeintrag / reserviert Speicherplatz

MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES MODIFY-FILE-GENERATION-SUPPORT	verändert Katalogeintrag / reserviert Speicherplatz oder gibt ihn frei
EDIT-FILE-ATTRIBUTES EDIT-FILE-GROUP-ATTRIBUTES EDIT-FILE-GENERATION-SUPPORT	aktiviert den geführten Dialog des entsprechenden MODIFY-Kommandos und ermöglicht das „Editieren“ eines bestehenden Katalogeintrags
IMPORT-FILE	erstellt Katalogeinträge für Dateien (importieren)
IMPORT-NODE-FILES	erstellt Katalogeinträge für Node-Files (importieren)
COMPARE-DISK-FILES	vergleicht zwei Plattendateien
COPY-FILE	kopiert Dateien
CONCATENATE-DISK-FILES	verkettet SAM-Dateien
DELETE-FILE DELETE-FILE-GROUP DELETE-FILE-GENERATION DELETE-SYSTEM-FILE	löscht Datei löscht FGG löscht Dateigeneration löscht Systemdatei
EXPORT-FILE	exportiert Katalogeinträge von Dateien auf privaten Datenträgern und Net-Storage-Volumes
EXPORT-NODE-FILES	exportiert Katalogeinträge von Node-Files (auf Net-Storage-Volumes)
LIST-NODE-FILES	informiert über Node-Files auf Net-Storage-Volumes
MAIL-FILE	sendet Datei oder Bibliothekselement per E-Mail an eine Benutzerkennung
SHOW-FILE-ATTRIBUTES	gibt Informationen aus dem Dateikatalog aus
CHECK-FILE-CONSISTENCY REPAIR-DISK-FILE	rekonstruiert beschädigte Dateien
REMOVE-FILE-ALLOCATION-LOCKS	hebt Dateisperre auf

REPAIR-FILE-LOCKS	beseitigt unberechtigt sitzende Dateisperren

2.2 Steuerung der ACS-Funktionen

Benötigte Dateien und auch Jobvariablen können mit Aliasnamen angesprochen werden. Die interne Zuordnung von Aliasname zu realem Namen wird in einem taskspezifischen Aliaskatalog verwaltet.
Außerdem kann ein Präfix definiert werden, das gegebenenfalls den Namen vorangestellt wird. Zur Steuerung der ACS-Funktionen stehen folgende Kommandos (eine Programmschnittstelle wird nicht angeboten) zur Verfügung:

Kommando	Kurzbeschreibung
ADD-ACS-SYSTEM-FILE	vereinbart eine ACS-Systemdatei
ADD-ALIAS-CATALOG-ENTRY	fügt einen Eintrag im Aliaskatalog der aktuellen Task hinzu und legt den Wirkungsbereich der Ersetzung fest
HOLD-ALIAS-SUBSTITUTION	unterbricht die ACS-Funktion für die laufende Task; es findet keine Namensersetzung mehr statt
LOAD-ALIAS-CATALOG	übernimmt gespeicherte Einträge aus einer AC-Datei in den Aliaskatalog
MODIFY-ACS-OPTIONS	ändert tasklokale Optionen
MODIFY-ALIAS-CATALOG-ENTRY	ändert bestehenden Eintrag im Aliaskatalog
PURGE-ALIAS-CATALOG	löscht den Alias-Katalog der laufenden Task; die eingestellten Optionen bleiben jedoch erhalten
REMOVE-ALIAS-CATALOG-ENTRY	löscht bestehenden Eintrag im Aliaskatalog
RESUME-ALIAS-SUBSTITUTION	nimmt die ACS-Funktion Namensersetzung wieder auf
SET-FILE-NAME-PREFIX	vereinbart Präfix für Datei-/JV-Namen und legt Wirkungsbereich des Präfix fest
SHOW-ACS-OPTIONS	informiert über tasklokale Optionen
SHOW-ACS-SYSTEM-FILES	informiert über vordefinierte Aliaskatalog-Systemdateien
SHOW-ALIAS-CATALOG-ENTRY	informiert über ausgewählte Aliaskatalog-Einträge (SYSOUT)
SHOW-FILE-NAME-PREFIX	zeigt aktuellen Präfix und seinen Gültigkeitsbereich an
STORE-ALIAS-CATALOG	speichert Aliaskatalog in Datei ab

2.3 Steuerung der Dateiverarbeitung

Für die Verarbeitung von Dateien durch ein Programm muss eine Verbindung zwischen Datei und Programm hergestellt werden können. Diese Verbindung kann im Programm bereits festgelegt oder erst vor Aufruf des Programms hergestellt werden, wenn das Programm mit einem programminternen Dateinamen (= Dateikettungsname) arbeitet. Die Verbindung wird neben anderen Informationen in der Task File Table (TFT) hinterlegt. Die Steuerung der Dateiverarbeitung erfolgt also über die TFT.

Für NK-ISAM-Dateien umfasst die Steuerung der Dateiverarbeitung außerdem die Verwaltung von Benutzer-ISAM-Pools, in denen diese Dateien verarbeitet werden. Es können auch Standard-ISAM-Pools des Systems genutzt werden, dann hat der Benutzer jedoch keinen Einfluss auf Poolgröße und -belegung.

Makros

Makro	Operanden	Kurzbeschreibung
ADDPLNK		weist einem Benutzer-ISAM-Pool einen Poolkettungsnamen zu
CHNGE		weist einer Datei einen neuen Dateikettungsnamen zu
CREAIX		erzeugt einen Sekundärindex für eine ISAM-Datei
CREPOOL		richtet einen Benutzer-ISAM-Pool ein
DELAIX		löscht Sekundärindizes einer ISAM-Datei
DELPOOL		löscht einen Benutzer-ISAM-Pool
DROPTFT		gibt einen gesperrten TFT-Eintrag frei
FCB	FILE	feste Zuordnung von Datei und Programm
	LINK	definiert den Dateikettungsnamen im Programm
	POOLLNK	stellt die Verbindung zum Benutzer-ISAM-Pool her
FILE	LINK	erstellt einen TFT-Eintrag, weitere Operanden beschreiben Datei- und Verarbeitungseigenschaften
	BLKCTRL	legt das Datenformat fest
	NFTYPE	legt den Dateityp einer Datei auf Net-Storage fest (BS2000-Datei oder Node-File)
	POOLLNK	stellt die Verbindung zum Benutzer-ISAM-Pool her
RDTFT		gibt TFT-Informationen aus
RELTFT		löscht einen TFT-Eintrag
REMPLNK		löscht den Poolkettungsnamen
SHOPLNK		informiert über die Zuordnung von ISAM-Pools zu Poolkettungsnamen
SHOPOOL		informiert über Attribute und Belegungszustände von ISAM-Pools

Kommandos

Kommando	Operanden	Kurzbeschreibung
ADD-FILE-LINK		erstellt einen TFT-Eintrag; weitere Operanden beschreiben Datei- und Verarbeitungseigenschaften
	BLOCK-CONTROL-INFO	legt das Dateiformat fest
	POOL-LINK	stellt die Verbindung zum ISAM-Pool her
EDIT-FILE-LINK		aktiviert den geführten Dialog des ADD-FILE-LINK-Kommandos und ermöglicht das „Editieren“ eines bestehenden TFT-Eintrags
SHOW-FILE-LINK		gibt TFT-Informationen aus
CHANGE-FILE-LINK		weist einer Datei einen neuen Dateikettungsnamen zu
LOCK-FILE-LINK		sperrt einen TFT-Eintrag, sodass ein REMOVE-FILE-LINK wirkungslos ist
UNLOCK-FILE-LINK		hebt die (LOCK-FILE-LINK-)Sperre für den TFT-Eintrag auf
REMOVE-FILE-LINK		löscht einen TFT-Eintrag
CREATE-ISAM-POOL		richtet einen Benutzer-ISAM-Pool ein
SHOW-ISAM-POOL-ATTRIBUTES		gibt Attribute und Belegungszustände von ISAM-Pools aus
DELETE-ISAM-POOL		löscht den Benutzer-ISAM-Pool
ADD-ISAM-POOL-LINK		weist einem Benutzer-ISAM-Pool einen Poolkettungsnamen zu
REMOVE-ISAM-POOL-LINK		löscht den Poolkettungsnamen
SHOW-ISAM-POOL-LINK		gibt die Zuordnung von ISAM-Pools zu Poolkettungsnamen aus
START-CACHING-FILES		Caching von bereits geöffneten Dateien starten
STOP-CACHING-FILES		Caching einzelner Dateien beenden

2.4 Unterstützung von Datenschutz und Datensicherheit

Die vom DVS automatisch unterstützten Mechanismen für Datei- und Datenschutz (Überprüfung der Zugriffsberechtigung usw.) können z.B. durch die Vergabe von Kennwörtern erweitert werden. Datensicherheit wird durch verschiedene Mechanismen zur Wiederherstellung von Dateien oder Programmen usw. gewährleistet.

2.4.1 Dateischutz

Makros

Makro	Operanden	Kurzbeschreibung
CATAL	SHARE	gibt Dateizugriff für fremde Benutzerkennungen frei
	ACCESS	bestimmt die zulässige Zugriffsart (Lesen/Schreiben)
	OWNERAR GROUPAR OTHERAR	legt in der BASIC-ACL die Zugriffsrechte für die Benutzergruppen fest
	GUARDS	Bei Einsatz von SECOS: Erweiterter Zugriffsschutz für Dateien
	EXPASS RDPASS WRPASS	Kennwörter für die verschiedenen Zugriffsebenen
	RETPD	legt eine Schreibschutzfrist fest
	PROTECT	Übernahme von Schutzattributen
CHKFAR		überprüft die Zugriffsrechte des Aufrufers auf Dateien
COPFILE	PROTECT	übernimmt beim Kopieren die Schutzmerkmale der Originaldatei
DECFILE		wandelt verschlüsselte Datei in unverschlüsselte Datei um
ENCFILE		wandelt unverschlüsselte Datei in verschlüsselte Datei um
FILE	RETPD	legt eine Schreibschutzfrist fest (nur in Zusammenhang mit Dateieröffnung!)
FCB	PASS	ermöglicht Zugriff bei Kennwortschutz
	RETPD	legt eine Schreibschutzfrist fest

Kommandos

Kommando	Operanden	Kurzbeschreibung
CREATE-FILE CREATE-FILE-GROUP MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES	USER-ACCESS	schränkt die Mehrbenutzbarkeit ein
	ACCESS	schränkt die Zugriffsart ein
	EXEC-PASSWORD READ-PASSWORD WRITE-PASSWORD	vergibt Kennwörter für die verschiedenen Zugriffsebenen
MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES	RETENTION-PERIOD	legt eine Schutzfrist fest

CREATE-FILE MODIFY-FILE-ATTRIBUTES	BASIC-ACL/GUARDS	setzt die Zugriffsrechte für die Datei
COPY-FILE	PROTECTION=SAME	übernimmt die Schutzmerkmale beim Kopieren einer Datei
	REPLACE-OLD-FILES	steuert, ob eine Datei mit dem Kommando COPY-FILE überschrieben werden kann
ADD-FILE-LINK	RETENTION-PERIOD	legt eine Schutzfrist fest (nur in Zusammenhang mit Dateieröffnung!)
ADD-PASSWORD		gibt ein Kennwort an, um den Zugriff auf kennwortgeschützte Dateien zu ermöglichen
REMOVE-PASSWORD		löscht ein Kennwort aus der Kennwortliste

2.4.2 Datenschutz

Makros

Makro	Operanden	Kurzbeschreibung
CATAL	DESTROY	legt im Katalogeintrag fest, dass Plattendateien beim Löschen überschrieben werden bzw. auf Bändern fremde Restdaten überschrieben werden (siehe FILE: DESTOC)
DECFILE		wandelt verschlüsselte Datei in unverschlüsselte Datei um
ENCFILE		wandelt unverschlüsselte Datei in verschlüsselte Datei um
ERASE	DESTROY	zerstört Daten beim Löschen
FILE	DESTOC	überschreibt beim Bandwechsel oder nach dem Schließen der Datei auf dem Band folgende (alte) Daten

Kommandos

Kommando	Operanden	Kurzbeschreibung
ADD-CRYPTO-PASSWORD		hinterlegt das Crypto-Kennwort zur Entschlüsselung von verschlüsselten Dateiinhalten in der Kennworttabelle der Task
ADD-FILE-LINK	DESTROY-OLD-CONTENTS	überschreibt bei Bandwechsel oder nach dem Schließen einer Banddatei auf dem Band vorhandene Restdaten
CREATE-FILE CREATE-FILE-GROUP MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GROUP-ATTRIBUTES	DESTROY-BY-DELETE	legt im Katalogeintrag fest, dass Daten auf Platten bei Speicherplatzfreigabe überschrieben werden bzw. auf Bändern fremde Restdaten überschrieben werden (siehe nächste Zeile: ADD-FILE-LINK DESTROY-OLD-CONTENTS)
DECRYPT-FILE		wandelt verschlüsselte Datei in unverschlüsselte Datei um
DELETE-FILE	DESTROY-ALL	zerstört Daten beim Löschen

DELETE-FILE-GROUP DELETE-FILE-GEN	DESTROY-ALL	zerstört Daten beim Löschen
ENCRYPT-FILE		wandelt unverschlüsselte Datei in verschlüsselte Datei um und definiert das zugehörige Crypto-Kennwort
REMOVE-CRYPTO-PASSWORD		entfernt das Crypto-Kennwort aus der Kennworttabelle der laufenden Task

2.4.3 Datensicherheit

Makros

Makro	Operanden	Kurzbeschreibung
CATAL	BACKUP	definiert die Häufigkeit der automatischen Sicherung
	NUM_OF_BACKUP_VERS	definiert, ob die Datei am HSMS Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im HSMS Versions-Backup-Archiv gesichert werden sollen
COPFILE	REPLACE	legt fest, ob eine existente Datei beim Kopieren überschrieben wird
CREPOOL	WROUT	sofortiges Rückschreiben bei Aktualisierung von ISAM-Dateien
FILE	WRCHK	Kontrollesen, um Aufzeichnungsfehler beim Schreiben auf Platte zu vermeiden
	WROUT	sofortiges Rückschreiben bei Aktualisierung von ISAM-Dateien
VERIF		rekonstruiert Dateistrukturen / entsperrt Dateien
WRCPT		schreibt einen Fixpunkt / erstellt eine Fixpunktdatei für den Wiederanlauf mit dem Kommando RESTART-PROGRAM
FCB	EXIT	Adresse einer Exit-Routine oder eines EXLST-Makroaufrufs
	WRCHK	Kontrollesen, um Aufzeichnungsfehler beim Schreiben auf Platte zu vermeiden
EXLST		definiert Exit-Routinen für Fehler und andere Ereignisse
LFFSNAP		listet Dateien von einem Snapset
LJFSNAP		listet Jobvariablen von einem Snapset
RFFSNAP		restauriert Dateien von einem Snapset
RJFSNAP		restauriert Jobvariablen von einem Snapset

Kommandos

Kommando	Operanden	Kurzbeschreibung
CREATE-FILE, CREATE-FILE-GROUP, MODIFY-FILE-ATTRIBUTES, MODIFY-FILE-GROUP-ATTRIBUTES	BACKUP-CLASS	Häufigkeit der automatischen Sicherung
CREATE-FILE, MODIFY-FILE-ATTRIBUTES	NUM-OF-BACKUP-VERS	definiert, ob die Datei am HSMS Versions-Backup teilnimmt bzw. wieviele Datei-Versionen maximal im HSMS Versions-Backup-Archiv gesichert werden sollen
COPY-FILE	REPLACE-OLD-FILES	legt fest, ob eine existente Datei beim Kopieren überschrieben wird
CREATE-ISAM-POOL	WRITE-IMMEDIATE	WROUT-Funktion bei SCOPE=TASK einschalten (sofortiges Zurückschreiben geänderter Blöcke)
	SCOPE=HOST-SYSTEM	WROUT-Funktion implizit eingeschaltet
ADD-FILE-LINK	SUPPORT=DISK (WRITE-CHECK)	Kontrolllesen geschriebener Sätze zur Vermeidung von Aufzeichnungsfehlern
	SUPPORT=DISK(ISAM-ATTR=*PAR(WRITE-IMMEDIATE))	sofortiges Rückschreiben bei Aktualisierung von ISAM-Dateien (bei SHARED-UPDATE=*NO)
RESTART-PROGRAM		Neustart eines abgebrochenen Programms, für das eine Fixpunktdatei erstellt wurde
SECURE-RESOURCE-ALLOCATION		reserviert Dateien oder Datenträger für einen Benutzerauftrag
CHECK-FILE-CONSISTENCY REPAIR-DISK-FILE		rekonstruiert Dateistrukturen
REMOVE-FILE-ALLOCATION-LOCKS		entsperrt Dateien
LIST-FILE-FROM-SNAPSET		listet Dateien von einem Snapset
LIST-JV-FROM-SNAPSET		listet Jobvariablen von einem Snapset

RESTORE-FILE-FROM-SNAPSET		restauriert Dateien von einem Snapset
RESTORE-JV-FROM-SNAPSET		restauriert Jobvariablen von einem Snapset

2.5 Geräte- und Datenträgerverwaltung

Das DVS unterstützt den Benutzer bei der Verarbeitung von Dateien auf privaten Datenträgern durch die Möglichkeit, Geräte und Datenträger für seinen Auftrag zu reservieren.

Makros

Makro	Operanden	Kurzbeschreibung
FILE	DEVICE VOLUME	bestimmt Gerätetyp und Datenträger für eine Datei auf privaten Datenträgern und Net-Storage-Volumes
	MOUNT	fordert die Bereitstellung privater Datenträger an der Konsole
IMPORT		erstellt Katalogeinträge für Dateien (importieren)
RELTFT		löscht TFT-Einträge und gibt implizit Geräte frei

Kommandos

Kommando	Operanden	Kurzbeschreibung
UNLOCK-FILE-LINK		hebt TFT-Sperre auf: vorangegangenes REMOVE-FILE-LINK kann wirksam werden und private Geräte freigeben
CREATE-FILE CREATE-FILE-GROUP CRE-FILE-GENERATION MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GENERATION-SUPPORT MODIFY-FILE-GROUP-ATTRIBUTES	DEVICE-TYPE/ VOLUME	legt Geräte und Datenträger für eine Datei fest
ADD-FILE-LINK	ADD-CATALOG-VOLUME	
CREATE-FILE CRE-FILE-GEN IMPORT-FILE MODIFY-FILE-GENERATION-SUPPORT MODIFY-FILE-ATTRIBUTES	SUPPORT=TAPE (PREMOUNT-LIST)	fordert die Bereitstellung privater Datenträger oder Net-Storage-Volumes an der Konsole an

MODIFY-FILE-ATTRIBUTES MODIFY-FILE-GENERATION-SUPPORT	SUPPORT=DISK (VOLUME-ALLOCATION)	
ADD-FILE-LINK	NUMBER-OF-PREMOUNTS	
LOCK-FILE-LINK		sperrt einen TFT-Eintrag; dadurch kein REMOVE-FILE-LINK mit Gerätefreigabe möglich
IMPORT-FILE		erstellt Katalogeinträge für Dateien (importieren)
REMOVE-FILE-LINK		löscht TFT-Einträge und gibt implizit Geräte frei
SECURE-RESOURCE-ALLOCATION		reserviert Geräte/Datenträger für einen Auftrag

2.6 Verwalten und Nutzen von Net-Storage

Das DVS unterstützt die Systembetreuung bei der Verwaltung von Net-Storage und den Benutzer bei der Bearbeitung von Dateien auf einem Net-Storage-Volume.

Makros

Makro	Operanden	Kurzbeschreibung
CATAL		erstellt oder ändert Katalogeinträge
ERASE		löscht Dateien / exportiert Dateien
FILE	DEVICE=NETSTOR, VOLUME=	bestimmt Gerätetyp und Datenträger für eine Datei auf einem Net-Storage-Volume
	NFTYPE=	bestimmt Dateityp einer Datei auf einem Net-Storage-Volume: BS2000-Datei oder Node-File
	STATE=FOREIGN	Datei von einem Net-Storage-Volume importieren
FSTAT	FROM=(vsn, NETSTOR)	Informationen aus dem Katalog des mit „vsn“ bezeichneten Net-Storage-Volumes
	STOTYPE=*NETSTOR	Nur Dateien, die auf Net-Storage-Volumes liegen, werden ausgewählt
	VTOC=*YES	Ausgabe der Katalogeinträge aus dem Katalog eines Net-Storage-Volumes
IMPORT	DEVICE=NETSTOR, VOLUME=	importiert Katalogeinträge von Dateien auf einem Net-Storage-Volume
RDTFT		Informationen über TFT-Einträge

Kommandos

Kommando	Operanden	Kurzbeschreibung
MOUNT-NET-STORAGE		Net-Storage verbinden
UMOUNT-NET-STORAGE		Net-Storage trennen
LIST-NET-DIRECTORIES		Net-Storage-Verzeichnisse auflisten
SHOW-NET-STORAGE		in BS20000 gemounteten Net-Storage anzeigen
ADD-NET-STORAGE-VOLUME		Net-Storage-Volume einrichten und einem lokalen Pubset zuordnen
REMOVE-NET-STORAGE-VOLUME		Net-Storage-Volume einem lokalen Pubset entziehen

SHOW-PUBSET-NET-STORAGE		Net-Storage eines Pubsets anzeigen
CREATE-FILE, MODIFY-FILE-ATTRIBUTES	STORAGE-TYPE= *NET-STORAGE, DEVICE-TYPE= NETSTOR, VOLUME=	Die Datei wird auf einem Net-Storage-Volume eingerichtet
	FILE-TYPE=	legt die Datei als BS2000-Datei oder Node-File an
SHOW-FILE-ATTRIBUTES	STORAGE-TYPE=*NET-STORAGE	wählt Dateien von Net-Storage aus
	FROM-CATALOG=*NET	gibt Katalogeinträge vom Net-Storage-Volume aus
	FILE-TYPE=	wählt nach Dateityp aus: BS2000-Datei oder Node-File
LIST-NODE-FILES		listet Node-Files auf Net-Storage-Volume auf
DELETE-FILE	STORAGE-TYPE=*NET-STORAGE	wählt Dateien von Net-Storage aus
	FILE-TYPE=	wählt nach Dateityp aus: BS2000-Datei oder Node-File
IMPORT-FILE, CHECK-IMPORT-DISK-FILE	DEVICE-TYPE=NETSTOR, VOLUME=	importiert/prüft Katalogeinträge von Dateien auf Net-Storage-Volumes
IMPORT-NODE-FILE		importiert Node-Files (von Net-Storage)
EXPORT-FILE	STORAGE-TYPE=*NET-STORAGE, VOLUME=	exportiert Dateien von Net-Storage
EXPORT-NODE-FILE		exportiert Node-Files
SHOW-FILE-LINK		gibt Informationen über TFT-Einträge aus
CHECK-FILE-CONSISTENCY, REMOVE-FILE-ALLOCATION-LOCKS, REPAIR-DISK-FILES	SELECT=*NET-STORAGE	wählt Dateien von Net-Storage aus

2.7 Zugriff zu Dateien

Der Dateizugriff erfolgt durch die Aktionsmakroaufrufe der verschiedenen Zugriffsmethoden. Das DVS übernimmt aber auch das Öffnen und Schließen einer Datei (OPEN/CLOSE-Behandlung) in Abhängigkeit von der jeweiligen Zugriffsart.

2.7.1 Dateiverarbeitung

Die DVS-Makros der Dateiverarbeitung (d.h. die „Service-Makros“) sind für alle Zugriffsmethoden gültige Makroaufrufe.

Makro	Kurzbeschreibung
FCB	legt einen Dateisteuerblock an (File Control Block = FCB)
FCBAD	legt den FCB im Literalbereich eines Programms an
OPEN	eröffnet eine Datei
CLOSE	schließt eine oder mehrere Dateien
EXLST	definiert Fehlerausgänge
EXRTN	Rücksprung aus EXLST-Routinen
LBRET	Rücksprung aus Benutzer-Kennsatzroutinen (Bandverarbeitung)

2.7.2 Zugriffsmethodenspezifische Makroaufrufe

Es werden folgende Zugriffsmethoden unterschieden:

- BTAM
- DIV
- EAM
- FASTPAM
- SAM
- ISAM
- UPAM

In den folgenden Tabellen sind die Makroaufrufe für den Dateizugriff den einzelnen Zugriffsmethoden zugeordnet.

UPAM = User Primary Access Method

Grundlage der Dateiverarbeitung ist der Standardblock (= PAM-Seite). UPAM unterstützt den blockorientierten Dateizugriff. Es werden ein oder mehrere logische Blöcke übertragen, oder auch Teile von logischen Blöcken. Mit UPAM lassen sich auch Dateien verarbeiten, die nicht mit UPAM erstellt wurden.

Makro	Kurzbeschreibung
PAM	steuert alle UPAM-Zugriffe
Bei ereignisgesteuerter Verarbeitung sind zusätzlich folgende Makroaufrufe von Bedeutung (ausführliche Beschreibung siehe Handbuch „Makroaufrufe an den Ablaufteil“ [2]):	
CHKEI	prüft die Warteschlangenbelegung einer Ereigniskennung
CONTXT	greift auf den Registersatz des unterbrochenen Prozesses zu
DISCO	schließt die Routine für den Contingency-Prozess
DISEI	trennt das Benutzerprogramm von der Ereigniskennung
ENACO	eröffnet eine Routine als Contingency-Prozess und weist ihr Namen und Priorität zu
ENAEI	richtet Ereigniskennung ein und/oder stellt Verbindung her zwischen aufrufendem Prozess und Ereigniskennung
FECB	richtet einen Steuerblock für die Ereigniskennung ein (= File Event Control Block)
LEVCO	ändert die Priorität des aufgerufenen Prozesses
POSSIG	signalisiert ein Ereignis
RETCO	beendet aufrufenden Contingency-Prozess
SOLSIG	fordert Signal von der Ereigniskennung an
SUSPEND	versetzt den aufrufenden Prozess in einen unterbrechbaren Wartezustand

FASTPAM = Fast Primary Access Method

Grundlage der Dateiverarbeitung ist der Standardblock (= PAM-Seite). FASTPAM unterstützt den blockorientierten Dateizugriff. Es werden ein oder mehrere logische Blöcke übertragen. Mit FASTPAM lassen sich auch Dateien verarbeiten, die nicht mit FASTPAM erstellt wurden.

Makro	Kurzbeschreibung
FPAMSRV	Verwaltungsaufrufe formulieren • Systemumgebung (FASTPAM-Environment) vorbereiten • Ein-/Ausgabebereiche (FASTPAM-IO-Area-Pool) vorbereiten • Datei zur Bearbeitung eröffnen • eine mit FPAMSRV geöffnete Datei schließen • Systemumgebung (FASTPAM-Environment) abbauen • Ein-/Ausgabebereiche (FASTPAM-IO-Area-Pool) abbauen
FPAMACC	Dateizugriffe formulieren • synchrones Lesen und Schreiben von logischen Blöcken • asynchrones Lesen und Schreiben von logischen Blöcken • Warten auf die Beendigung asynchroner Ein-/Ausgabeaufträge • Benachrichtigung über die Beendigung asynchroner Ein-/Ausgabeaufträge

DIV = Data In Virtual

Mit DIV lassen sich Dateien direkt in einen virtuellen Adressraum abbilden. Die Übertragung erfolgt „on demand“, angestoßen durch einen Page-fault bei einem Zugriff auf eine Speicherseite in einem „Fenster“. Mit DIV lassen sich auch Dateien verarbeiten, die nicht mit DIV erstellt wurden.

Makro	Kurzbeschreibung
DIV	Dateien mit der Zugriffsmethode DIV bearbeiten • Datei eröffnen • Fenster (Arbeitsbereich im virtuellen Adressraum) definieren • Modifizierte Seiten aus dem Fenster in die Datei auf Platte zurückschreiben • Änderungen im Fenster zurücknehmen • Fenster im virtuellen Adressraum freigeben • Datei schließen; ggf. noch existierende Fenster werden mit Standardwerten freigegeben.

SAM = Sequential Access Method

Eine SAM-Datei ist eine Folge von Sätzen (, die in logischen Blöcken enthalten sind).
Das DVS ermöglicht Ihnen die Verarbeitung aufeinander folgender Datensätze, und zwar sowohl in Richtung Dateiende als auch in Richtung Dateianfang. SAM erfüllt für die Bandverarbeitung alle von DIN 66029 bis Austauschstufe 3 gestellten Anforderungen; es können sowohl Dateien mit Standardblöcken als auch mit Nichtstandardblöcken verarbeitet werden.

Makro	Kurzbeschreibung
FEOV	löst Bandwechsel aus
GET	stellt den nächsten Satz bereit
PUT	schreibt den nächsten Satz an das bisherige Dateiende
PUTX	(nur im Ortungsbetrieb) ersetzt einen durch GET positionierten Satz im Puffer
RELSE	schließt einen Datenblock ab
SETL	positioniert auf Dateianfang, -ende oder auf einen Satz, der mit GET bereitgestellt werden soll

BTAM = Basic Tape Access Method

BTAM ist eine Zugriffsmethode für blockorientierte Bandverarbeitung; mit BTAM lassen sich auch Banddateien verarbeiten, die nicht mit BTAM erstellt wurden. Während der Verarbeitung einer Banddatei kann die Verarbeitungsrichtung innerhalb der Datei beliebig gewechselt werden, Bänder können beliebig block- oder abschnittsweise positioniert werden. BTAM verarbeitet Dateien mit oder ohne Standardblockung.

Makro	Kurzbeschreibung
BTAM	steuert alle BTAM-Aktionen
FEOV	löst einen Bandwechsel aus
NDWERINF	fragt Status-Bytes ab

EAM = Evanescent Access Method

Mit EAM werden temporäre Dateien im SYSEAM-Bereich verarbeitet. EAM ist eine blockorientierte Zugriffsmethode und eignet sich vor allem für die schnelle Verarbeitung von auftragsabhängigen Arbeitsdateien.

Makro	Kurzbeschreibung
EAM	steuert alle EAM-Zugriffe

ISAM = Indexed Sequential Access Method

Eine ISAM-Datei besteht aus einer Menge von Sätzen. Jeder Satz enthält ein Schlüsselfeld, über das er direkt adressiert werden kann. Die Sätze sind logisch nach aufsteigenden Schlüsseln geordnet; beim sequenziellen Zugriff werden sie in dieser Reihenfolge geliefert.

Makro	Kurzbeschreibung

ELIM	streicht einen Satz aus der Datei
GET	liest den in der Datei folgenden Satz (sequenzielles Lesen)
GETFL	bei Verwendung von markierten ISAM-Schlüsseln: liest den folgenden Satz innerhalb des Markierungsbereichs (sequenziell)
GETKY	liest den ersten Satz mit dem angegebenen Schlüssel
GETR	liest den vorhergehenden Satz (sequenzielles Lesen, rückwärts)
INSRT	fügt einen Satz mit neuem ISAM-Schlüssel in die Datei ein
ISREQ	hebt eine ISAM-Sperre auf
OSTAT	informiert über Anzahl und Art gleichzeitiger Dateizugriffe
PUT	schreibt sequenziell Sätze an das Dateiende (inklusive Prüfung der Schlüssel auf gültige Reihenfolge)
PUTX	ersetzt einen durch GET o.Ä. bereitgestellten Satz
RETRY	setzt nach Durchlaufen des EXLST-PGLOCK-Ausgangs den ISAM-Zeiger neu und wiederholt den letzten Makroaufruf
SETL	positioniert den ISAM-Zeiger für anschließende sequenzielle Verarbeitung auf Dateianfang, -ende oder einen bestimmten Satz
STORE	fügt einen Satz mit neuem ISAM-Schlüssel in die Datei ein überschreibt einen Satz mit bereits vorhandenem ISAM-Schlüssel, wenn Mehrfachvergabe von ISAM-Schlüsseln nicht zulässig ist fügt einen Satz mit bereits vorhandenem ISAM-Schlüssel als letzten Satz mit diesem Schlüssel in die Datei ein

Die ISAM-Pool-Makroaufrufe sind auf "Steuerung der Dateiverarbeitung" aufgeführt.

2.8 Generierung von Operandenlisten für Steuerblöcke, DVS-Tabellen usw.

Der Benutzer kann an der Programmschnittstelle bestimmte Programmbereiche oder DUMMY-Sections (DSECTs) erzeugen, die es ihm ermöglichen, auf Inhalte von DVS-Tabellen, Dateisteuerblöcken usw. oder auf die Operandenlisten von DVS-Makros mit symbolischen Adressen zuzugreifen.
Für die meisten Makroaufrufe ist dies möglich über den MF-Operanden, anderenfalls gibt es spezielle DSECT-Makroaufrufe. Bei „älteren“ Makroaufrufen (z.B. CATAL), die erst in einer späteren Version umgestellt wurden, entscheidet der VERSION-Operand, ob der spezielle DSECT-Makroaufruf weiter genutzt werden muss oder ob die Angabe über den MF-Operanden möglich ist.

Steuerblöcke und zugriffsmethoden-spezifische Makroaufrufe

Makro	Kurzbeschreibung
IDBPL	Operandenliste für den BTAM-Makroaufruf
IDECB	für den UPAM-Ereigniskennung-Steuerblock (FECB)
IDFCB	für den Dateisteuerblock (FCB) des Benutzerprogramms auf der TU-Ebene
IDFCBE	Erweiterung des 24-Bit-TU-FCB
IDMCB	EAM-Steuerblock für EAM-Makroaufruf
IDPPL	Operandenliste für den PAM-Makroaufruf
IDOST	Operandenliste für den OSTAT-Makroaufruf

DVS-Tabellen, Dateikatalog usw.

Makro	Kurzbeschreibung
IDCE	Katalogeintrag
IDCEG	Katalogeintrag (Zusatz für Dateigenerationsgruppen)
IDCEX	Katalogeintrag (Erweiterung)
IDEE	Katalogeintrag (Extentliste)
IDEMS	DVS-Fehlermeldungen
IDTFT	TFT-Eintrag
IDVT	Eintrag des Datenträger-Kennsatzes (Volume Table)

2.9 Ausgabe von Informationen über Dateien, Datenträger, Geräte usw.

Mit verschiedenen Makros und Kommandos lassen sich Informationen über Katalogeinträge und Zustand von Dateien, die Task File Table, die Belegung von Geräten und Datenträgern usw. anfordern und für die weitere Verarbeitung auswerten.

Makros

Makro	Kurzbeschreibung
FSTAT	Informationen aus dem Dateikatalog bzw. über Katalogeinträge von Dateien
RDTFT	Informationen über TFT-Einträge
OSTAT	Informationen über Zahl und Art der Zugriffe verschiedener Aufträge auf eine ISAM-Datei
SHOPOOL	Informationen über ISAM-Pools
SHOPLNK	Informationen über ISAM-Pool-Kettungsnamen
SHOWAIX	Informationen über Sekundärschlüssel einer ISAM-Datei

Kommandos

Kommando	Kurzbeschreibung
SHOW-FILE	Ausgabe einer Datei auf SYSOUT
SHOW-FILE-ATTRIBUTES	Informationen aus dem Datei-Katalog bzw. über Katalogeinträge von Dateien
SHOW-FILE-LINK	Informationen über TFT-Einträge
SHOW-FILE-LOCKS	Informationen über Dateisperren
SHOW-DEVICE-STATUS	Informationen über die vom Benutzer-Auftrag online belegten Geräte
SHOW-DISK-DEFAULTS	Informationen über die vom Operator festgelegten Standardwerte für Privatplatten
SHOW-INDEX-ATTRIBUTES	Informationen über bestehende Sekundär-Indizes einer NK-ISAM-Datei
SHOW-ISAM-POOL-ATTRIBUTES	Informationen über Attribute und Belegungszustände von NK-ISAM-Pools
SHOW-ISAM-POOL-LINKS	Informationen über die Zuordnung von NK-ISAM-Pools zu Pool-Link-Namen (Pool-Kettungsnamen)
SHOW-MOUNT-PARAMETER	Informationen über die vom Operator eingestellten Vorgaben zum Montieren und Demontieren von Datenträgern

SHOW-PUBSET-FILE-SERVICES	Informationen über das Service-Angebot eines SM-Pubsets
SHOW-RESOURCE-ALLOCATION	Informationen über Geräte- und Datenträgerbelegungen sowie offene Operator-Aktionen für den anfordernden Auftrag
SHOW-NET-STORAGE	in BS20000 gemounteten Net-Storage anzeigen
SHOW-PUBSET-NET-STORAGE	Net-Storage eines Pubsets anzeigen

3 Datenträger

Im BS2000 gibt es gemeinschaftliche und private Datenträger sowie Net-Storage. Nur Platten können gemeinschaftliche Datenträger sein.
Net-Storage gehört nicht zu den gemeinschaftlichen Datenträgern, dient jedoch zu deren Erweiterung. Net-Storage wird im BS2000 hinsichtlich des Gerätetyps zu der Gruppe der Platten gerechnet.
Private Datenträger sind alle Bänder und alle nicht als gemeinschaftlich gekennzeichneten Platten. Beide Arten von Datenträgern können nebeneinander benutzt werden.

Bei der Definition von Platten unterscheidet das DVS gemeinschaftliche Platten (Public Volumes), die zu Pubsets zusammengestellt werden, und private Platten (Private Disks). Net-Storage-Volumes werden als Erweiterung zu Pubsets genutzt.

Alle Datenträger werden im BS2000 durch einen Namen identifiziert, der bis zu sechs Zeichen lang sein darf. Dieser Name wird als VSN (Volume Serial Number) oder auch Archivnummer bezeichnet. Der gültige Zeichenvorrat besteht aus den alphanumerischen Zeichen A..Z, 0..9 und den Sonderzeichen Punkt, @, # und $. Pubset-Volumes sind von anderen Datenträgern durch ihre VSN unterscheidbar, wobei die VSN gemeinschaftlicher Platten einer Konvention unterliegt: Sie muss mit der Zeichenkette „PUB“ beginnen oder einen Punkt an der dritten, vierten oder fünften Stelle enthalten. Diese VSN-Syntax darf für private Datenträger und Net-Storage-Volumes nicht angewandt werden.

Das DVS übernimmt die Speicherplatzverwaltung auf Platten. Einige Kontrollfunktionen (Zugriffsberechtigung, Sättigungssteuerung, usw.) sind nur auf einem Pubset anwendbar und nicht auf privaten Platten oder auf Net-Storage.

3.1 Gemeinschaftliche Platten (Pubsets)

Gemeinschaftliche Platten werden zu Volume-Sets oder SF-Pubsets zusammengefasst. Ein oder mehrere Volume-Sets bilden einen SM-Pubset.
Mit MPVS (Multiple Public Volume Set) können mehrere Pubsets in einem Systemlauf gleichzeitig installiert sein. An einen Pubset können Net-Storage-Volumes angehängt werden. Sie sind aber nicht Bestandteil des Pubsets (IMPORT-PUBSET ist auch ohne gemountete Net-Storage-Volumes möglich).

Es gibt einen ausgezeichneten Pubset, den „Home-Pubset“, der während des gesamten Systemlaufs verfügbar sein muss, da er die zum Laden, Betreiben und Beenden des Systems benötigten Daten enthält. Die anderen Pubsets können von der Systembetreuung importiert und exportiert werden. Die Default-Pubsets der Benutzer sollten immer importiert werden.

Alle Datenträger eines Pubsets werden vom System als eine Einheit betrachtet und verwaltet. Jeder Benutzer, der dazu durch die Systembetreuung über das Kommando ADD-USER bzw. durch den Systemparameter FSHARING berechtigt ist, kann auf importierten Pubsets mit allen DVS-Funktionen Dateien erzeugen, verarbeiten und löschen.

Es gibt zwei verschiedene Typen von Pubsets: den SF-Pubset (Single Feature Pubset) und den SM-Pubset (System Managed Pubset). Sie unterscheiden sich durch ihre Eigenschaften, Betriebsparameter und Service-Attribute.

SF-Pubsets

Ein SF-Pubset besteht aus Volumes, die die gleichen physikalischen Eigenschaften aufweisen. Er repräsentiert ein begrenztes, in sich homogenes Service-Angebot.

SM-Pubsets

Ein SM-Pubset besteht aus einem oder mehreren Volume-Sets. Die Volume-Eigenschaften müssen nur innerhalb eines Volume-Sets homogen sein. Eine Datei kann sich nicht über mehrere Volume-Sets erstrecken. Jeder Volume-Set repräsentiert anhand seiner Eigenschaften einen speziellen Service-Typ innerhalb eines SM-Pubsets, der bei der Auswahl des Ablageortes einer Datei berücksichtigt wird. Grundlage für die Auswahl sind die vom Benutzer über die Angabe von logischen Dateiattributen angeforderten Services.

Betriebsarten von Pubsets

Shared-Pubset

Bei Einsatz des Produkts MSCF (Mehrrechnersystem) und einer entsprechenden Hardware-Konfiguration ist der gleichzeitige und gemeinsame Zugriff über mehrere BS2000-Systeme hinweg auf einen Pubset möglich.
Maximal 16 BS2000-Systeme, die in einem gemeinsamen MSCF-Verbund gekoppelt werden, können über einen direkten Hardware-Pfad als „Sharer“ auf diesen mehrbenutzbaren Pubset zugreifen. Einer dieser Verbund-Teilnehmer wird zum temporären Eigentümer dieses Pubsets ernannt und wickelt für die anderen Systeme die Funktionen zur Verwaltung der Dateien, der Benutzer und der Zugriffe ab. Alle Verwaltungs-Anforderungen seitens der untergeordneten Teilnehmer, der sog. „Slave-Sharer“, müssen über MSCF an das Eigentümersystem, den „Master“ gerichtet werden.
Bei Ausfall des Eigentümersystems wird an allen abhängigen Systemen eine pubsetspezifische Jobvariable gesetzt. Wurde von der Systembetreuung über das Kommando SET-PUBSET-ATTRIBUTES ein Backup-Eigentümer definiert, kann dieser Slave-Sharer dann die Rolle des ausgefallenen Master-Sharer übernehmen (Master-Wechsel).
Ist kein Backup-Eigentümer vorgesehen, kann die Systembetreuung am Slave-Sharer den Pubset exportieren und bei einem nachfolgenden IMPORT-PUBSET-Kommando einen neuen Eigentümer bestimmen.

Das gesamte Konzept des Shared-Pubset (Hardware-Konfiguration, Verwaltung der Pubsets, Datenzugriffe) ist ausführlich im Handbuch „HIPLEX MSCF“ [11] beschrieben.

XCS-Pubset

Mit HIPLEX MSCF steht eine erweiterte Verbundfunktionalität zur Verfügung: der XCS-Verbund (Cross Coupled System), der eine engere Koordination der beteiligten Systeme bietet. Jedes System hat eine konsistente und vollständige Sicht des gesamten Verbunds. Der XCS-Verbund bietet damit Mechanismen zur Realisierung verteilter Anwendungen.
Er ist in erster Linie als Verfügbarkeits- und Lastverbund des BS2000 konzipiert. Dem Benutzer werden u.a. folgende, im DVS-Umfeld wichtige Funktionen angeboten:

- Distributed Lock Manager (DLM):
  Diese Funktion realisiert eine system-übergreifende Sperrenverwaltung und unterstützt damit system-übergreifende Synchronisation und Serialisierung. Sie ist Basisfunktion für SFS.
- Shared File System (SFS):
  Das SFS erlaubt innerhalb des XCS-Verbunds die system-übergreifende Aktualisierung von Dateien auf Shared-Pubsets, die nicht notwendig XCS-Pubsets sein müssen. In HIPLEX MSCF wird dieser globale Shared-Update für die block- bzw. bytestrom-orientierten Zugriffsmethoden UPAM, FASTPAM und DIV unterstützt.

Voraussetzungen für einen XCS-Verbund sind:

- ein System kann max. einem XCS-Verbund angehören
- die Teilnehmer müssen voll vermascht sein, d.h. es müssen MSCF-Verbindungen zwischen allen Systemen des Verbunds bestehen
- dem XCS-Verbund muss mindestens ein XCS-Pubset angehören, zu dem von allen Systemen aus Zugriffspfade vorhanden sein müssen

Ein XCS-Pubset dient als zentraler Ablageort für verbundweit benötigte Daten. XCS-Pubsets werden automatisch durch das System importiert.

Das gesamte Konzept des XCS-Verbunds (Funktionalität, Generierung und Betrieb) ist ausführlich im Handbuch „HIPLEX MSCF“ [11] beschrieben.

Mit SHC-OSD erstellte Pubset-Kopien

Für die Datensicherung von Pubsets in Plattenspeichersystemen können mit Hilfe von SHC-OSD auf dem jeweiligen Plattenspeichersystem spezielle Pubset-Kopien (Spiegelplatten) lokal erstellt werden. Es gibt zwei Ausführungen:

- Pubset-Kopie, die auf Basis von Clone-Units erstellt wurde (z.B. für die Sicherung (Backup) eines konsistenten Stands des Pubsets mit HSMS oder FDDRL)
- Pubset-Kopie, die auf Basis von Snap-Units erstellt wurde, auch Snapset genannt (siehe Abschnitt „Pubset-Sicherung mit Snapsets“)

Die Pubset-Kopien sind dem Pubset fest zugeordnet. Die VSNs ihrer Volumes genügen einer speziellen Syntax (siehe „Spezielle VSN-Benennung bei Pubset-Kopien“ im nachfolgenden Abschnitt). Auf diese Kopien ist nur lesender Zugriff möglich. Für Einzelheiten siehe die Handbücher „HSMS“ [10] und „Systembetreuung“ [7].

Namensgebung von Pubsets

Die Namen von Platten, die einem Volume-Set/SF-Pubset zugeordnet sind, müssen syntaktisch zum Namen des Volume-Sets/SF-Pubsets passen. Dabei ist zwischen ein- und mehrstelligen Volume-Set- bzw. Pubset-Namen zu unterscheiden:

- für einstellige Namen gibt es die PUB-Notation
- für zwei- bis vierstellige Namen gibt es die Punkt-Notation

Standard-Namen von Net-Storage-Volumes, die einem Pubset zugeordnet sind, haben, abhängig von der Pubset-Notation, eine besondere Form, siehe „Notation von Net-Storage- Volumes“ im Handbuch „Systembetreuung“ [7].

VSN in PUB-Notation

Diese Art der Pubset-Adressierung hat ihren Namen aus der Verwendung der Zeichenkette „PUB“ als festen, unveränderlichen Bestandteil der VSN. Dabei weist „PUB“ (public) darauf hin, dass es sich um gemeinschaftliche Platten handelt.

Eine VSN in PUB-Notation besteht immer aus 6 Zeichen und hat folgendes Format:

```
PUBpxx
```

`PUB`	unveränderlicher Bestandteil zur Unterscheidung von privaten Platten (3 Zeichen „PUB“) = Typbezeichner
`p`	Katalogkennung (Catid), (1 Zeichen; A..Z, 0..9)
`xx`	laufende Nummer innerhalb eines Pubsets/Volume-Sets, (2 Zeichen; 00..31) = Folgenummer

Mit der einstelligen Katalogkennung können maximal 36 Pubsets bzw. Volume-Sets adressiert werden, die aus bis zu 32 Platten bestehen können.

Beispiele: PUBA00, PUBA25, PUB502

VSN in Punkt-Notation

Der Name dieser Notation ist durch die Verwendung eines Punktes als Trennungszeichen zwischen der Katalogkennung und der Folgenummer im Pubset bzw. Volume-Set entstanden. Auch die Punkt-Notation bezeichnet immer gemeinschaftliche Platten.

Eine VSN in Punkt-Notation besteht immer aus 6 Zeichen und hat folgendes Format:

```
pp[pp].[xy]z
```

`pp[pp]`	Katalogkennung (Catid), (2-4 Zeichen; je A..Z, 0..9), der Präfix „PUB“ ist nicht erlaubt
`.`	Punkt (1 Zeichen) = Typbezeichner
`[xy]z`	laufende Nummer innerhalb eines Pubsets/Volume-Sets, (1-3 Zeichen; je 0..9) = Folgenummer

Ein Pubset bzw. Volume-Set in Punkt-Notation kann aus bis zu 255 Platten bestehen.

Beispiele: AA.001, AB.309, XYZ.23, OTTO.0, J19P.8

Die Katalogkennung bei SF-Pubsets korrespondiert direkt mit dem Namensteil „Katalogkennung“ in der VSN. Bei SM-Pubsets ist die Katalogkennung verschieden von allen Volume-Set-Namen dieses SM-Pubsets, und damit auch verschieden vom VSN-Namensteil „Katalogkennung“ aller Platten des Pubsets.

Spezielle VSN-Benennung bei Pubset-Kopien

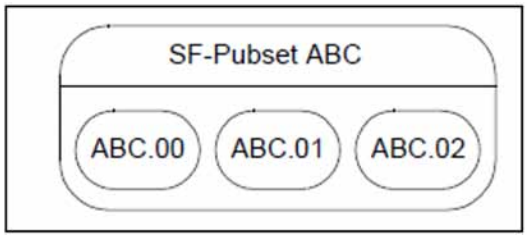
Die VSNs einer mit SHC-OSD auf Basis von Clone-Units erstellten Pubset-Kopie werden aus den Original-VSNs gebildet, wobei bei der PUB-Notation das „U“ der Zeichenfolge „PUB“ bzw. bei der Punkt-Notation der Punkt durch einen Doppelpunkt ersetzt wird.

Die VSNs einer mit SHC-OSD auf Basis von Snap-Units erstellten Pubset-Kopie, d.h. eines Snapsets werden nach folgender Regel aus den Original-VSNs gebildet:

- Bis zum 26. Snapset (d.h. für die Snap-Ids a bis z):
  - Bei der PUB-Notation werden das „U“ und das „B“ der Zeichenfolge „PUB“ jeweils durch die Snapset-Id in Kleinbuchstaben ersetzt.
  - Bei der Punkt-Notation wird der Punkt durch die Snapset-Id in Kleinbuchstaben ersetzt.
- Ab dem 27. Snapset (d.h. für die Snap-Ids A bis Z):
  - Bei der PUB-Notation werden das „P“ und das „U“ der Zeichenfolge „PUB“ jeweils durch die Snapset-Id in Kleinbuchstaben ersetzt.
  - Bei der Punkt-Notation wird der Punkt durch die Snapset-Id in Kleinbuchstaben ersetzt und gleichzeitig seine Zeichenposition verändert:
    - Bei 4-stelliger Catid wird die Snapset-Id nach dem ersten Zeichen der Catid eingefügt.
    - Bei 3-stelliger Catid wird die Snapset-Id vor dem ersten Zeichen der Catid eingefügt.
    - Bei 2-stelliger Catid wird die Snapset-Id als letztes Zeichen der VSN eingefügt.

Beispiele

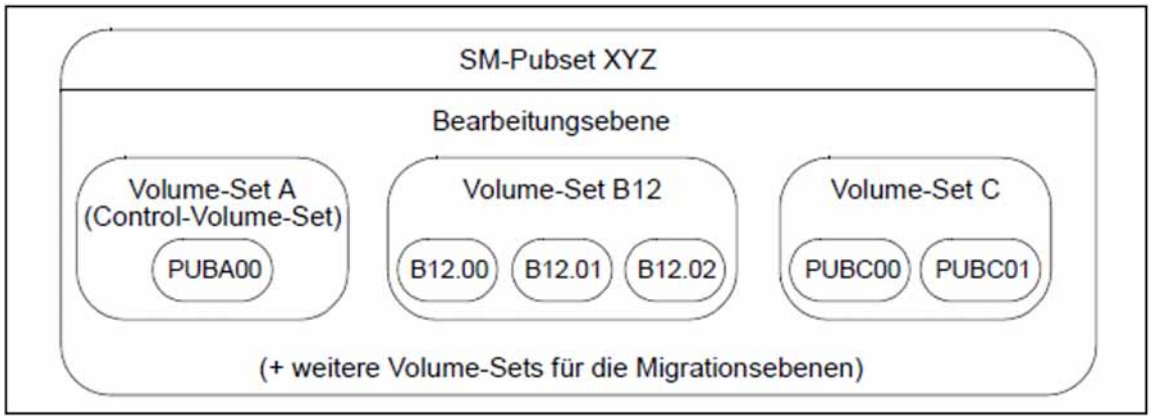
Datenträgerkonfiguration eines SF-Pubsets:



Der SF-Pubset mit der Katalogkennung `ABC` besteht aus drei Platten.
Existiert eine Datei mit dem Dateinamen „MEINE.LISTE“ unter der Benutzerkennung „ALLERLEI“ auf einer der Platten des SF-Pubsets ABC, so lautet ihr Pfadname:
„:ABC:$ALLERLEI.MEINE.LISTE“

Ein zugehöriger Snapset mit der Snapset-Id x würde dann aus den Volumes mit den VSNs `ABCx00`, `ABCx01` und `ABCx02` bestehen.

Datenträgerkonfiguration eines SM-Pubsets:



Der SM-Pubset mit der Katalogkennung XYZ besteht aus drei Volume-Sets.
Existiert eine Datei mit dem Dateinamen „TELEFONLISTE“ unter der Benutzerkennung „EINERLEI“ auf einer der Platten, die zu einem der Volume-Sets des SM-Pubsets XYZ zusammengefasst sind, so lautet ihr Pfadname: „:XYZ:$EINERLEI.TELEFONLISTE“, d.h. für die Adressierung ist es nicht relevant, wie der SM-Pubset intern aufgebaut ist und wo genau seine Datei vom System abgelegt wurde.

Ein zugehöriger Snapset mit der Snapset-Id `m` würde dann aus den Volumes mit den VSNs `PmBA00`, `B12m00`, `B12m01`, `B12m02`, `PmBC00` und `PmBC01` bestehen.

Benutzerkatalog

Jeder Pubset enthält einen Benutzerkatalog und seinen eigenen Dateikatalog.

Durch Einträge im Benutzerkatalog ist geregelt, wer bzw. welche Benutzerkennung (Userid) auf einen Pubset zugreifen kann. Benutzer ohne Eintrag im Benutzerkatalog für den Pubset können nicht auf diesen Pubset zugreifen, auch nicht auf mehrbenutzbare Dateien, es sei denn, dies ist durch den Systemparameter FSHARING generell erlaubt.

Im Benutzerkatalog sind auch für jeden Benutzer, der darin eingetragen ist, pubset-spezifische Berechtigungen festgelegt. Enthält der Eintrag im Benutzerkatalog z.B. die Angabe PUBLIC-SPACE-LIMIT=0, kann der Benutzer auf diesem Pubset keine permanenten Dateien anlegen; er kann jedoch auf alle mehrbenutzbaren Dateien, die in diesem Pubset katalogisiert sind, zugreifen.
Das Anlegen von temporären Dateien kann durch die Angabe TEMP-SPACE-LIMIT=0 verhindert sein (zur Speicherplatzverwaltung siehe „Anfordern von Speicherplatz“).
Die Angabe NET-STORAGE-USAGE=*ALLOWED legt fest, dass der Benutzer Speicherplatz auf Net-Storage belegen darf.

Mit dem Kommando SHOW-USER-ATTRIBUTES kann sich der Benutzer über seinen Eintrag im Benutzerkatalog informieren.

3.2 Privatplatten

Privatplatten müssen im Gegensatz zu gemeinschaftlichen Platten für die Dateiverarbeitung separat angefordert werden. Sie können von einem Auftrag exklusiv oder von mehreren gleichzeitig laufenden Aufträgen gemeinsam benutzt werden.
Die Betriebsart ist normalerweise „shareable“.
Mit dem Kommando SECURE-RESOURCE-ALLOCATION kann eine exklusive Nutzung für den Benutzerauftrag eingestellt werden. Der Operator kann über spezielle Kommandos die einzelnen Belegungsmodi sperren.

Die Dateiverarbeitung auf Privatplatten verläuft genauso wie die von Dateien in MPVS-Umgebung.

Wenn ohne Fern-Katalogzugriff (RCA) auf eine Datei zugegriffen werden soll, die von einem anderen System erzeugt wurde, muss diese Datei zuvor dem eigenen System (TSOSCAT) durch den Makro FILE oder das Kommando IMPORT-FILE bekannt gemacht werden.

Mit dem Operanden VTOC=YES/NO im Makro FSTAT bzw. mit dem Operanden INFORMATION im Kommando IMPORT-FILE kann die Ausgabe der VTOC-Katalogeinträge der privaten Platten an Stelle der TSOSCAT-Einträge veranlasst werden. Dabei ersetzt der VTOC-Eintrag des privaten Datenträgers den entsprechenden TSOSCAT-Eintrag.

3.3 Net-Storage

BS2000 ermöglicht den Zugang zu UNIX-Dateisystemen über NFS.
Dateien von BS2000 und von offenen Systemen können damit im Server-Netzwerk auf dem freigegebenen Speicherplatz von File-Servern abgelegt und bearbeitet werden.

Folgende Begriffe werden in BS2000 beim Arbeiten mit Net-Storage verwendet:

Net-Server

File-Server im weltweiten Rechnernetz, der Speicherplatz (Network Attached Storage, NAS) für die Nutzung durch andere Server bereitstellt und entsprechende File-Server-Dienste anbietet.

Net-Storage

Der von einem Net-Server im Rechnernetz bereitgestellte und zur Nutzung durch fremde Server freigegebene Speicherplatz. Net-Storage kann ein Dateisystem oder auch nur ein Knoten im Dateisystem des Net-Servers sein.

Net-Client

Realisiert den Zugriff auf Net-Storage für das nutzende Betriebssystem.
In BS2000 transformiert der Net-Client zusammen mit dem BS2000-Subsystem ONETSTOR die BS2000-Dateizugriffe in entsprechende UNIX-Dateizugriffe und führt sie über NFS auf dem Net-Server aus.

Net-Client bei SU /390 und S-Server ist der HNC, bei SU x86 das Trägersystem X2000.

Net-Storage-Volume

Net-Storage-Volumes repräsentieren Net-Storage im BS2000, die die Systembetreuung als Erweiterung von Daten-Pubsets bereitstellt.
Net-Storage-Volumes werden durch ihre Volume Serial Number (VSN) und den Volumetyp NETSTOR angesprochen. Die VSN des Net-Storage-Volumes entspricht im freigegebenen Dateisystem des Net-Servers dem Verzeichnis, das die Benutzerdateien und Metadaten enthält.

> ¡ Die Bereitstellung und Verwaltung von Net-Storage ist Aufgabe der Systembetreuung, siehe das Handbuch „Einführung in die Systembetreuung“ [7].

Net-Storage-Datei

Bezeichnet eine Datei, die auf einem Net-Storage-Volume angelegt ist. Auf Net-Storage wird zwischen den zwei Dateitypen BS2000-Datei und Node-File unterschieden.

BS2000-Datei

Bezeichnet eine Datei, die ausschließlich von BS2000 angelegt und bearbeitet wird. BS2000-Dateien auf Net-Storage (FILE-TYPE=BS2000) werden seit BS2000/OSD-BC V9.0 bedient. Sie liegen direkt auf einem Net-Storage-Volume. Offene Systeme dürfen ausschließlich lesend darauf zugreifen.

Node-File

Bezeichnet eine Net-Storage-Datei (FILE-TYPE=NODE-FILE), die sowohl von BS2000 als auch von offenen Systemen angelegt und bearbeitet werden kann. Node-Files werden ab BS2000 OSD/BC V10.0 bedient. Sie liegen auf einem Net-Storage-Volume in einem benutzerspezifischen Verzeichnis (Name der Benutzerkennung) und die Dateinamen entsprechen den BS2000-Namenskonventionen.

Interoperabilität auf Net-Storage

Sowohl BS2000 als auch offene Systeme können sich mit demselben Net-Storage verbinden, dort Node-Files anlegen und diese wechselseitig bearbeiten:

- BS2000 kann Node-Files auf Net-Storage anlegen und bearbeiten - offene Systeme können diese Node-Files ebenfalls bearbeiten
- Offene Systeme können Node-Files auf Net-Storage anlegen und bearbeiten - BS2000 kann diese Node-Files importieren und ebenfalls bearbeiten

3.3.1 Zugriff von BS2000 auf Net-Storage

Der Zugriff auf eine Datei auf Net-Storage durch den BS2000-Anwender erfolgt über DVS-Schnittstellen und den Net-Client auf dem Net-Server in folgenden Schritten:

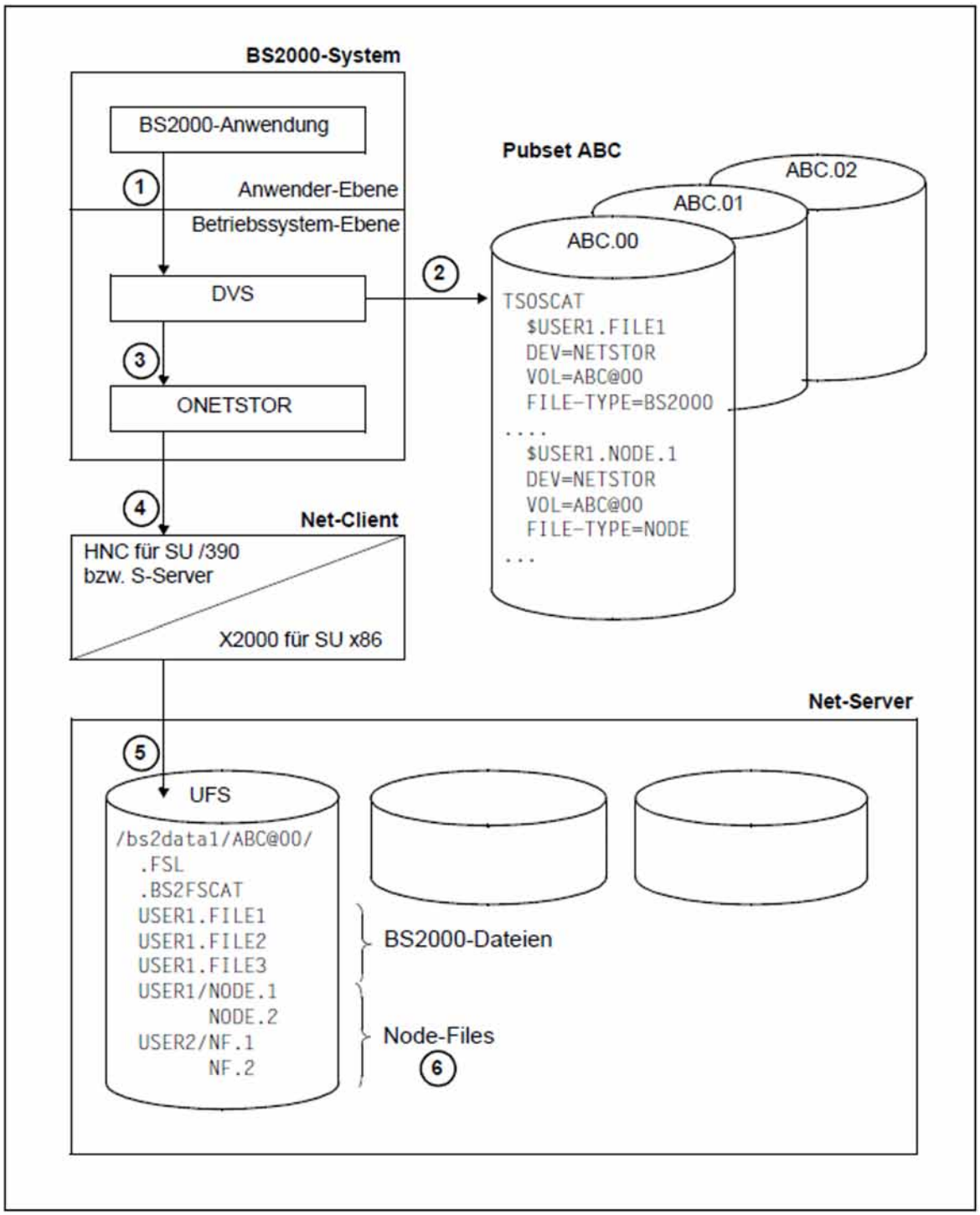


Bild 1: BS2000-Zugriff auf Net-Storage

1. Eine BS2000-Anwendung unter der Benutzerkennung `USER1` möchte auf die Datei `FILE1`, die auf Net-Storage liegt und im Pubset ABC katalogisiert ist, zugreifen. Dies geschieht über die üblichen Benutzer-Schnittstellen des DVS, siehe Abschnitt „Arbeiten mit Net-Storage“.

2. Das DVS prüft die Existenz der Datei im Benutzer- und Dateikatalog des Pubsets ABC. Anhand der Dateiattribute DEVICE=NETSTOR und VOLUME=ABC@00 erkennt das DVS, dass es sich um eine Datei im Verzeichnis ABC@00 auf dem vom Net-Server freigegebenen Net-Storage handelt.
3. Der eigentliche Zugriff auf die Datei FILE1 erfolgt über das BS2000-Subsystem ONETSTOR und den Net-Client.
4. Das BS2000-Subsystem ONETSTOR transformiert den BS2000-Dateizugriff in den entsprechenden Dateizugriff im UNIX-Dateisystem (UFS) und reicht ihn an den Net-Client weiter.
5. Am Net-Client ist das vom Net-Server freigegebenen Verzeichnis /bs2data1/ (mit Unterverzeichnis ABC@00) eingehängt. Der Dateizugriff erfolgt über NFS im UFS des Net-Servers.
6. Beim Zugriff auf ein Node-File, z.B. die Datei NODE.1, erkennt das DVS, dass es sich um ein Node-File (FILE-TYPE=NODE) im Verzeichnis ABC@00 auf dem vom Net-Server freigegebenen Net-Storage handelt. Node-Files liegen in einem benutzerspezifischen Verzeichnis, das den Namen der Benutzerkennung trägt, hier das Verzeichnis USER1. Dieses Verzeichnis wird automatisch angelegt, wenn BS2000 das erste Node-File des Benutzers anlegt.
   Damit ein Benutzer mit Node-Files arbeiten kann, müssen im Benutzereintrag des Pubsets die POSIX-Benutzerattribute Benutzernummer und Gruppennummer eingetragen sein. Für den Benutzer USER1 ist z.B. 1005 und 100 eingetragen (siehe Bild 2 (Zugriff offener Systeme auf Net-Storage)).

3.3.2 Zugriff offener Systeme auf Net-Storage

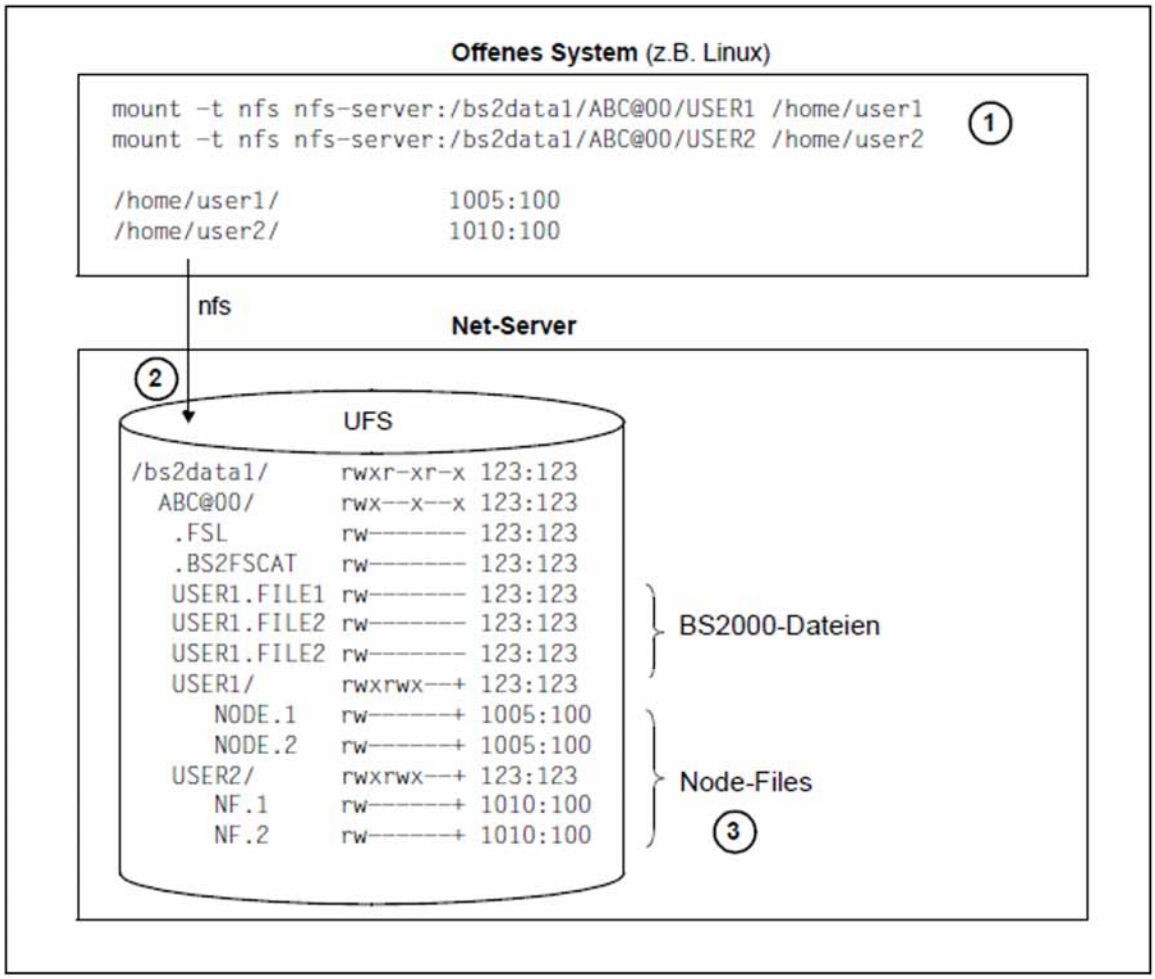


Bild 2: Zugriff offener Systeme auf Net-Storage

1. Offene Systeme erhalten Zugang zu BS2000-Dateien und Node-Files auf dem Net-Server durch entsprechende Mount-Kommandos.
2. Beim Anlegen der benutzerspezifischen Verzeichnisse werden von BS2000 auch die POSIX-ACLs entsprechend versorgt. Unter Linux erhalten deshalb im Beispiel oben die Benutzer `user1` und `user2` Zugriff auf das Verzeichnis USER1 und USER2.
3. Damit ein Benutzer sowohl von BS2000 als auch vom offenen System aus lesenden und schreibenden Zugriff hat, muss für ihn auf beiden Systemen dieselbe UserId und GroupId eingetragen sein. Das Gleiche gilt für das Anlegen und Löschen von Dateien.

> **i** Mit dem Einsatz von NFSv4 muss dies über einen Verzeichnisdienst, z.B. openLDAP oder Active Directory, gewährleistet werden, an den sowohl der Net-Client als auch der NFS-Server und das offenen System angeschlossen sind.

3.3.3 Arbeiten mit Net-Storage

BS2000-Anwender können Net-Storage-Volumes nach ihrer Einrichtung über die Schnittstellen des DVS nutzen, wenn dies in ihrem Benutzereintrag erlaubt ist. Der verfügbare Speicherplatz auf Net-Storage ist dabei nur durch die Größe des freigegebenen Speicherbereichs auf dem Net-Server begrenzt.

- Mit den DVS-Kommandos können Sie über den Operanden STORAGE-TYPE Dateien auf Net-Storage-Volumes einrichten und bearbeiten, z.B. mit

  ```
  /CREATE-FILE or /MODIFY-FILE-ATTRIBUTES
                  FILENAME=...,
                  SUPPORT=*PUBLIC-DISK(STORAGE-TYPE=*NET-STORAGE,..)
  ```

  Es werden Katalogeinträge sowohl im lokalen Pubset als auch im Katalog des Net-Storage-Volumes erzeugt. Die Dateien werden auch im UNIX-Filesystem mit der Dateigröße 0 angelegt. Siehe auch die „Randbedingungen“ .

  Das System ermittelt das Net-Storage-Volume, auf dem die Datei angelegt wird, wie folgt:

  - Gibt es zu dem unter FILE-NAME spezifizierten Pubset ein Standard-Net-Storage-Volume, wird automatisch dieses ausgewählt.
  - Gibt es kein Standard-Net-Storage-Volume zu dem spezifizierten Pubset, wird ein Nicht-Standard-Net-Storage-Volume ausgewählt. Falls mehrere Nicht-Standard-Net-Storage-Volumes existieren, ermittelt das System selbst ein Volume.

  Deshalb muss der Benutzer ein Volume angeben, wenn er von mehreren vorhandenen ein bestimmtes Nicht-Standard-Net-Storage-Volume auswählen möchte:

  ```
  /CREATE-FILE FILE-NAME=...,SUPPORT=*PUBLIC-DISK(STORAGE-CLASS=*NONE(
                VOLUME=NETS00, DEVICE=NETSTOR))
  ```

  Der Benutzer kann sich mit dem Kommando SHOW-PUBSET-NET-STORAGE Informationen über die einem Pubset zugeordneten Net-Storage-Volumes verschaffen.

  Ohne Angabe des Dateityps wird per Default eine BS2000-Datei angelegt. Um ein Node-File anzulegen, müssen Sie explizit den Dateityp angeben, z.B. mit

  ```
  /CREATE-FILE FILE-NAME=...,
               SUPPORT=*PUBLIC-DISK(
                  STORAGE-TYPE=*NET-STORAGE(FILE-TYPE=*NODE-FILE),
                      STORAGE-CLASS=*NONE(VOLUME=NET001,DEVICE=NETSTOR))
  ```

  Dabei wird, falls nicht bereits vorhanden, ein benutzerspezifisches Verzeichnis erzeugt, in dem die angegebene Datei angelegt wird. Das Verzeichnis erhält zusätzlich die notwendigen POSIX-ACLs, um dem Benutzer auch von offenen Systemen aus den Zugriff zu gewährleisten.

- Das Kommando SHOW-FILE-ATTRIBUTES zeigt Dateieigenschaften und den Speicherort (Volume und Gerätetyp NETSTOR) der Datei an. Standardmäßig werden die Informationen aus dem TSOSCAT des zugehörigen Pubsets angezeigt. Informationen aus dem Katalog eines Net-Storage-Volumes können Sie folgendermaßen anfordern:

  ```
  /SHOW-FILE-ATTRIBUTES ...,
          SELECT=*BY-ATTRIBUTES(FROM-CATALOG=*NET(VOLUME=...))
  ```

- Mit den DVS-Kommandos können Sie über den Auswahloperanden STORAGE-TYPE Dateien auf Net-Storage-Volumes auswählen, z.B. mit

  ```
  /SHOW-FILE-ATTRIBUTES oder /DELETE-FILE
          SELECT=*BY-ATTRIBUTES(STORAGE-TYPE=*NET-STORAGE,..,FILE-TYPE=...)
  ```

  Mit dem Operanden FILE-TYPE können Sie die Auswahl auf BS2000-Dateien oder auf Node-Files beschränken.

- BS2000-Dateien, die in BS2000 noch nicht bekannt sind, können Sie vom Net-Storage importieren, d.h. einen Katalogeintrag im lokalen Pubset erstellen, z.B. mit:

  ```
  /IMPORT-FILE SUPPORT=*DISK(VOLUME=P@BA00,DEVICE-TYPE=NETSTOR,...,PUBSET=A)
  ```

  Das Net-Storage-Volume muss dabei dem Pubset zugeordnet sein, in dessen Katalog die Einträge importiert werden sollen. Ein noch nicht zugeordnetes Net-Storage-Volume kann die Systembetreuung mit ADD-NET-STORAGE-VOLUME zuordnen.

- Sie können den Import auch vorab simulieren, z.B. mit:

  ```
  /CHECK-IMPORT-DISK-FILE VOLUME=P@BA00,DEVICE-TYPE=NETSTOR,PUBSET=A,...
  ```

- Node-Files, die in BS2000 noch nicht bekannt sind, können Sie vom Net-Storage importieren, d.h. einen Katalogeintrag im lokalen Pubset erstellen, z.B. mit:

  ```
  /IMPORT-NODE-FILE VOLUME=P@BA00, FILE-NAME=...[,FILE-STRUCTURE=...]
  ```

  Der Katalogeintrag wird aus den Inode-Eigenschaften des Node-File erzeugt. Für den Operanden FILE-STRUCTURE ist *STD voreingestellt. Dies entspricht der Angabe *PAM, womit die Datei als PAM-Datei importiert wird (Dateiattribut FILE-STRUC=PAM). Node-Files, die binäre Daten enthalten (z.B. Typ data, tar, executable, usw.), sollten als PAM-Datei importiert werden. Node-Files, die textbasierte Daten enthalten (Typ text), sollten mit der expliziten Angabe von FILE-STRUCTURE=*SAM als SAM-Datei importiert werden (Dateiattribut FILE-STRUC=SAM und REC-FORM=V). Details zur Verarbeitung von SAM-Node-Files siehe Abschnitt „Arbeiten mit SAM-Node-Files“.

  Importiert werden können nur „normale“ Dateien, falls ihre Namen den BS2000-Konventionen entsprechen.

- Die Informationen über die Node-Files auf einem Net-Storage-Volume können Sie unabhängig davon abfragen, ob sie in BS2000 bereits bekannt sind. Es werden aber nur Node-Files angezeigt, die den BS2000-Namenskonventionen entsprechen, d.h. Dateien, die auch importiert werden können. Informationen über die eigenen Node-Files erhalten Sie z.B. mit

  ```
  /LIST-NODE-FILES VOLUME=P@BA00, USER-DIRECTORY=*OWN, NODE-FILE-NAME=...
  ```

- Dateien auf Net-Storage-Volumes können Sie über den Operanden STORAGE-TYPE exportieren, z.B. mit

  ```
  /EXPORT-FILE ...,SELECT=*BY-ATTRIBUTES(STORAGE-TYPE=*NET-STORAGE,...)
  ```

  Node-Files können Sie mit /EXPORT-NODE-FILE exportieren.

  > **i** Das Exportieren einzelner Dateien ist bei der manuellen Behebung von Inkonsistenzen (z.B. nach Ausfall des Net-Storage) sinnvoll.

- Die Kommandos CHECK-FILE-CONSISTENCY, REMOVE-FILE-ALLOCATION-LOCKS und REPAIR-DISK-FILES unterstützen den Betrieb von Net-Storage über den Auswahloperanden SELECT=*NET-STORAGE.

- Auch die DMS-Programmschnittstellen bedienen Net-Storage, z.B. die Makros CATAL, ERASE, FILE, FSTAT, IMPORT und RDTFT.
  Der Zugriff auf eine Datei auf einem Net-Storage-Volume erfolgt in der herkömmlichen Weise durch Angabe von Katalogkennung, Benutzerkennung und Dateiname.

- Mit den BS2000-Softwareprodukten HSMS und ARCHIVE können Sie Dateien auf Net-Storage wie lokale Dateien sichern und wieder rekonstruieren, siehe das Handbuch „HSMS“ [10].

- Dateien auf Net-Storage können im UNIX- oder Windows-System des Net-Servers direkt verwendet werden, z.B. BS2000-Dateien als Steuer- oder Druckdateien:
  - Das Kommando CONVERT-FILE-TO-PDF erzeugt PDF-Dateien in einem binärkompatiblen Format.
  - Das Softwareprodukt BS2ZIP erzeugt Zip-Archive in einem WINZIP-kompatiblen Format.

  > ¡ Linux- oder Windows-Anwendungen dürfen auf BS2000-Dateien nur lesend zugreifen. Wenn Linux- oder Windows-Anwendungen auch schreibend auf die Dateien zugreifen sollen, sind Node-Files zu verwenden.

3.3.4 Arbeiten mit SAM-Node-Files

Während PAM-Dateien aus BS2000-Sicht strukturlos sind und sich zur Speicherung von beliebigen binären Inhalten eigenen, bestehen SAM-Dateien aus einer sequenziellen Folge von Datensätzen, die sich zur Speicherung von textbasierten Inhalten eignen. Um den Austausch von textbasierten Daten zwischen BS2000 und Systemen der offenen Welt zu ermöglichen, können SAM-Dateien als SAM-Node-Files auf Net-Storage abgelegt werden.

3.3.4.1 SAM-Blockstruktur im BS2000

Im BS2000 werden SAM-Dateien mit der satzorientierte Zugriffsmethode SAM sequentiell verarbeitet. Im Kontext von Net-Storage und Node-Files ist nur die SAM-Verarbeitung auf Platte relevant, d.h. Besonderheiten der Bandverarbeitung spielen in diesem Kontext keine Rolle. Als SAM-Node-Files auf Net-Storage werden nur NK-SAM-Dateien im Format BLKCTRL=DATA und mit variablem Satzformat (RECFORM=V) unterstützt.

Bei NK-SAM-Dateien mit diesen Dateiattributen erstellt die Zugriffsmethode SAM Datenblöcke mit folgender Struktur:

- Der logische Block, der aus n PAM-Pages (oder logischen Halbseiten LHP, (STD,n) mit n = 1..16) besteht, beginnt mit einem 12 Byte langen BLKCTRL-Feld und einem 4 Byte langen Datenlängenfeld. Das Datenlängenfeld gibt die Länge der Nutzdaten im logischen Block an.
- Darauf folgen die SAM-Datensätze, die jeweils aus Satzlängenfeld und Daten bestehen. Die Daten hinter den Nutzdaten bis zum Ende des logischen Blocks sind undefiniert. Beim sequentiellen Schreiben der Datei wird ein logischer Block satzweise gefüllt (Makro PUT), bis ein Satz nicht mehr in den aktuellen Block passt. Dieser Satz wird dann in den nächsten logischen Block geschrieben.
- Eine Anwendung kann aber auch den Datenblock früher „abschließen“ bevor er voll ist und einen neuen Datenblock beginnen (Makro RELSE).

Beim Lesen solcher NK-SAM-Dateien erwartet die Zugriffsmethode SAM die Datenblöcke ebenfalls in dieser Struktur. Für das Lesen der Datenblöcke gilt:

- Sequentielles Lesen erfolgt mit GET. Den aktuellen Block schließen und den nächsten Block einlesen erfolgt mit RELSE und anschließendem GET.
- Innerhalb der Datei kann mit SETL auf einen bestimmten Block und Satz positioniert werden.

Details zur Zugriffsmethode siehe Kapitel „SAM – Sequential Access Method“ (SAM - Sequential Access Method).

3.3.4.2 SAM-Konverter

Gegenüber der von SAM erwarteten Datenstruktur sind textbasierte Daten von Node-Files, abgesehen von Zeilenvorschubzeichen (Line Feed, LF), die einzelne Datensätze von einander trennen, strukturlos. Die Zugriffsmethode SAM benötigt deshalb einen „Übersetzer“, den SAM-Konverter, der die SAM-spezifische Verarbeitung einer BS2000-Plattendatei auf die Node-File-Verarbeitung abbildet. Der SAM-Konverter ist Bestandteil des Net-Client.

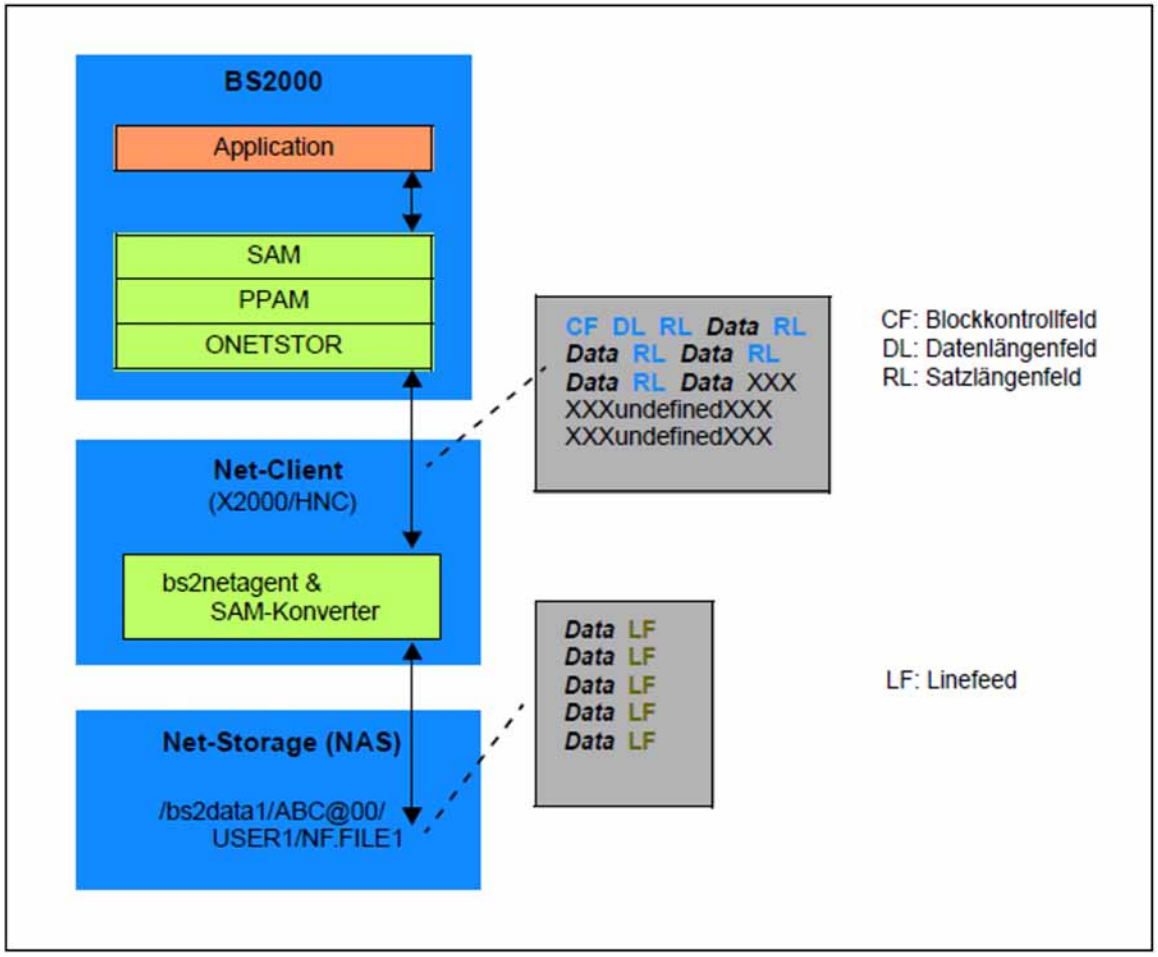


Bild 3: SAM-Node-File konvertieren

Der SAM-Konverter modifiziert einen logischen SAM-Block während des Schreibens bzw. Übertragens von BS2000 auf Net-Storage wie folgt:

- Das Blockkontrollfeld (CF), das Datenlängenfeld (DL) und die Satzlängenfelder (RL) werden entfernt.
- In Abhängigkeit vom Dateiattribut NET-CODED-CHAR-SET wird für die Daten eine Code-Konvertierung vorgenommen (siehe Abschnitt „Zeichensatz für Node-Files“).
- Nach jedem Datensatz wird ein zum Coded-Character-Set (CCS) passender Zeilenvorschub (LF) eingefügt.

Umgekehrt baut der SAM-Konverter beim Lesen der Daten logische SAM-Datenblöcke so auf, als ob die Zugriffsmethode SAM die Daten von einer BS2000-Platte lesen würde:

- Zu Beginn jedes logischen Blockes wird das Blockkontrollfeld und das Datenlängenfeld eingefügt.
- Die Datensätze aus dem Node-File werden um Satzlängenfelder erweitert.

Bei jedem OPEN analysiert der SAM-Konverter das SAM-Node-File und erstellt eine Node-File-Positionierungsliste, in der die Zuordnung der virtuellen logischen BS2000-Blocknummern zu den realen Byte-Positionen innerhalb der Datei auf dem Net-Storage vermerkt werden. Dabei wird gleichzeitig die Dateigröße in PAM-Seiten (LPP) ermittelt und an BS2000 zurückgemeldet. Mit Hilfe dieser Liste kann der SAM-Konverter beliebige logische Blöcke der Datei lesen und aufbereitet an BS2000 senden (z.B. Anzeige eines bestimmten Satzes bei SHOW-FILE). Diese Liste ist auch Voraussetzung für einen OPEN EXTEND oder REVERSE.

Die Positionierungsliste existiert pro geöffneter Datei temporär bis zum CLOSE. Bei sehr großen Dateien kann der Aufbau der Liste eine gewisse Zeit erfordern und auch größere Hauptspeicher-Ressourcen des Net-Clients in Anspruch nehmen. Wenn extrem große und viele Dateien gleichzeitig verarbeitet werden, kann es am Net-Client zu Ressourcen-Engpässen kommen, die mit entsprechenden Returncodes beantwortet werden.

Hinweis zur Verwendung von RELSE

Der Aufruf von RELSE schließt den aktuellen logischen Block und startet einen neuen logischen Block. Die Blocknummer der Wiedergewinnungsadresse wird um 1 erhöht und die Satznummer wird auf 0 zurückgesetzt. Die Positionierungsliste des Net-Client erhält dabei einen weiteren Eintrag, so dass diese Stelle wie bei der SAM-Verarbeitung auf Public-Space als Beginn eines logischen Blocks später wieder gefunden werden kann (SETL).

Bei SAM-Node-Files werden die Daten aber tatsächlich nicht blockorientiert sondern lückenlos hintereinander geschrieben. Sobald bei der Verarbeitung eines SAM-Node-Files mindestens ein RELSE verwendet wird, sind nach Schließen und erneutem Öffnen der Datei die gespeicherten Wiedergewinnungsadressen unbrauchbar, da sie ggf. von der Node-File-Positionierungsliste abweichen.

3.3.4.3 Zeichensatz für Node-Files

Für BS2000-Dateien beschreibt das CODED-CHARACTER-SET (CCS) den im BS2000 verwendeten Zeichensatz (siehe Abschnitt „Dateiattribute für Ausgabe von Dateien“). Für die Codierung in SAM-Node-Files werden in der Regel davon abweichende Zeichensätze verwendet. Damit eine Code-Konvertierung für die Bearbeitung und Darstellung von Node-Files im BS2000 möglich ist, wird ab BS2000 OSD/BC V11.0 für Node-Files zusätzlich der auf Net-Storage verwendete Zeichensatz im Dateikatalog eingetragen.

Das Dateiattribut NET-CODED-CHAR-SET (NETCCS) beschreibt den tatsächlichen Zeichensatz, mit dem die Daten eines Node-Files auf dem Net-Storage abgelegt werden. Bei der Darstellung bzw. bei der Bearbeitung der Datei im BS2000 werden EBCDIC-Zeichensätze entsprechend dieser Einstellung umgesetzt. Bei PAM-Node-Files findet keine Konvertierung statt.

Das NETCCS eines Node-File wird beim Erstellen des Katalogeintrags mit CREATE-FILE oder IMPORT-NODE-FILE bestimmt und kann mit MODIFY-FILE-ATTRBUTES geändert werden. Beim CREATE-FILE oder IMPORT-NODE-FILE werden die Zeichensätze CCS und NETCCS standardmäßig aus den Einstellungen im Benutzereintrag ermittelt. Alternativ kann der Benutzer beim CREATE-FILE für CCS und NETCCS abweichende Angaben machen. Mit der Angabe NET-CODED-CHAR-SET=*NO-CONV oder *ISO wird automatisch ein Zielzeichensatz ermittelt und in den Dateiattributen eingetragen. Das NET-CODED-CHAR-SET kann aber auch wie das CODED-CHARACTER-SET explizit angegeben werden. Die folgende Tabelle gibt einen Überblick.

In den ersten beiden Spalten seien die Einstellungen im Benutzereintrag bzw. die Angabe beim CREATE-FILE vorgegeben, aus denen das resultierende NET-CODED-CHAR-SET im Katalogeintrag des Node-Files ermittelt wird (siehe rechte Spalte):

Benutzereintrag CCS [1)]	**Benutzereintrag NETCSS** [1)]	**Resultierendes NETCSS im Katalogeintrag des Node-Files**
EDF03IRV/*NONE	*ISO	ISO88591; bei der Code-Umsetzung wird für CCS EDF041 angenommen
EDF03DRV	*ISO	ISO88591; bei der Code-Umsetzung wird für CCS EDF04DRV angenommen
EDF04DRV	*ISO	ISO88591
EDF04x	*ISO	ISO8859x mit x=1,2,..F
ISO8859x	*ISO oder *NO-CONV	ISOx
UTFx	*ISO oder *NO-CONV	UTFx
<name_a 1..8>	<name_b 1..8>	<name_b 1..8>
<name_a 1..8>	*NO-CONV	<name_a 1..8>

[1)] Benutzereintrag (SYSSRPM) bzw. Angabe im CREATE-FILE oder MODIFY-FILE-ATTRIBUTES

Hinweise:

- Die Angaben des Benutzers werden nicht validiert. Erst beim OPEN wird XHCS aufgerufen, um die benötigte Code-Tabelle abzurufen. Wenn diese dort nicht verfügbar ist, wird der OPEN abgewiesen.

- Das Ändern der Coded-Character-Sets (CCS bzw. NETCCS) bewirkt zunächst nur eine Änderung der entsprechenden Dateiattribute. Erst beim Verarbeiten der Datei (Lesen und Schreiben) kommen die Code-Tabellen beim Übertragen zum/vom Net-Storage und gleichzeitigen Konvertieren der Daten zur Anwendung.
- Die benötigten Zeichensätze sind unbedingt vor der Verarbeitung der Datei einzustellen. Nachdem Daten auf Net-Storage geschrieben wurden, sollten die Einstellungen für CCS und NETCCS nicht mehr geändert werden, da im Allgemeinen davon ausgegangen werden muss, dass die Umsetzung der Zeichen bei der weiteren Verarbeitung mit nunmehr geänderten Code-Tabellen zu Inkonsistenzen führen.
- Ab BS2000 OSD/BC V11.0 angelegte SAM- und PAM-Node-Files besitzen immer ein definiertes NETCCS. PAM-Node-Files, die vor V11.0 angelegt wurden, werden so behandelt, als ob sie mit der Angabe *NO-CONVERSION erstellt wurden.
- Relevant ist das NETCCS nur bei der Verarbeitung von SAM-Node-Files.
- Unabhängig von der Einstellung des Dateiattributs wird beim Schreiben des Node-Files zwischen jeden umgesetzten Satz ein zum Zielzeichensatz passendes Zeilenvorschub-Steuerzeichen eingefügt. Dies ist z.B. x'0A' (für ISO88591, ISO646, UTF-8), x'000A' (für UTF-16) oder x'15' (für EBCDIC).
- Auch wenn der Anwender als CCS einen 7-Bit Zeichensatz EDF03IRV oder EDF03DRV angegeben hat, wird trotzdem bei der Umsetzung der Zeichen für das Coded-Character-Set ein 8-Bit Zeichensatz verwendet: Bei EDF03IRV wird EDF041 und bei EDF03DRV wird EDF04DRV angenommen.
- Wenn für das CCS ein ISO- oder UTF-Zeichensatz angegeben ist, wird keine Konvertierung durchgeführt.
- Eine Umsetzung von 7- oder 8-Bit Zeichensätzen in einen Mehrbyte-Zeichensatz (z.B. UTF) ist nicht möglich.

3.3.5 Randbedingungen

Für das Arbeiten mit Net-Storage sind folgende Randbedingungen zu beachten:

- Eine Datei auf Net-Storage besteht nur aus einem Extent. Die Angabe einer Volume-Liste wird akzeptiert, es wird aber nur das erste Volume verwendet.
- Eine Speicherplatzbelegung im Operanden SPACE der Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES kann zwar angegeben werden, es wird aber kein Speicherplatz in dieser Größe auf Net-Storage belegt. Die (maximale) Speicherplatzbelegung findet erst dann statt, wenn die Datei in der entsprechenden Größe gespeichert wird.
- Ein Net-Storage-Volume verhält sich in Bezug auf die Allokierung wie eine NK2-Platte mit minimaler Allokierungseinheit 8 KB.
- Absolutzuweisungen sind für Dateien auf Net-Storage nicht möglich.
- Dateien mit folgenden Eigenschaften können **nicht** auf einem Net-Storage-Volume eingerichtet werden:
  - Dateien mit PAM-Schlüssel
  - Arbeitsdateien
  - temporäre Dateien
  - Dateigenerationsgruppen

- Für Node-Files gilt:
  - PAM-Node-Files werden ab BS2000 OSD/BC V10.0 unterstützt, SAM-Node-Files ab BS2000 OSD/BC V11.0. Weitere Voraussetzungen wie Hardware-Abhängigkeiten sind den aktuellen Freigabemitteilungen zu entnehmen.
  - Aus BS2000-Sicht können nur folgende Dateien Node-Files werden:
    - PAM-Dateien ohne PAM-Schlüssel (BLK-CONTROL=NO)
    - SAM-Dateien ohne PAM-Schlüssel (BLK-CONTROL=DATA) und variablem Satzformat (REC-FORM=V)
  - Node-Files sind strukturlos und enden auf Byte-Grenze.
  - Node-Files müssen für die Bearbeitung im BS2000 importiert sein.
  - Node-Files haben im UFS für den entsprechenden Benutzer die Zugriffsrechte:

    ```
    rw- --- ---
    ```

    Dies sind die für BS2000 minimal notwendigen Zugriffsrechte für den entsprechenden Benutzer. Ein Benutzer des offenen Systems kann diese Zugriffsrechte ändern. Ein BS2000-Benutzer kann dies nicht.
    Werden von Seiten der offenen Systeme die Eigentümerschaft oder die Zugriffsrechte (chown, chmod) geändert, kann BS2000 gegebenenfalls auf die Dateien nicht mehr zugreifen.
  - Node-Files können von BS2000-Seite nur als normale Dateien angelegt bzw. verwaltet werden. Zum Beispiel können Links nicht ins BS2000 importiert werden, da Besonderheiten bei der Verarbeitung von UNIX-Links (Hard- oder Softlinks) in BS2000 nicht abgebildet werden.
  - Wenn Node-Files als SAM-Datei importiert werden, ist Folgendes zu beachten:
    - Die Dateigröße wird zunächst aus dem Dateisystem in den Katalogeintrag übernommen. Dieser Wert ist aus BS2000-Sicht etwas zu klein, da die SAM-Struktur (Blockkontroll- und Längenfelder sowie ungenutzte Bytes am Blockende) noch nicht berücksichtigt ist. Der richtige Wert wird erst beim Öffnen der Datei ermittelt und dann im Katalog aktualisiert.
    - Für die Datei wird die maximale Blockgröße von (STD,16) im Katalog eingetragen. Zur Berechnung der maximal möglichen Datensatzlänge müssen von diesem Wert die Länge des Blockkontrollfeldes und des Satzlängenfeldes abgezogen werden. Damit kann BS2000 Sätze bis maximal 32768-20=32748 Bytes aus dem Node-File einlesen. Bei einem kleineren Blockungsfaktor reduziert sich die maximal verarbeitbare Satzlänge entsprechend. Deshalb wird empfohlen, für SAM-Node-Files grundsätzlich die maximale Blockgröße zu verwenden.
  - Beim Kopieren (COPY-FILE) von Node-Files ist Folgendes zu beachten:
    - Wenn in ein Node-File kopiert werden soll, muss für das Node-File bereits ein entsprechender Katalogeintrag existieren. Gegebenenfalls ist vorher ein Katalogeintrag mit dem Kommando CREATE-FILE zu erstellen:

      ```
      /CREATE-FILE <node-file>,SUPPORT=*PUBLIC-DISK(
                   STORAGE-TYPE=*NET-STORAGE(FILE-TYPE=*NODE))
      ```
    - Wenn eine SAM-Datei in ein Node-File kopiert wird, kann der Benutzer das gewünschte NETCSS beim Anlegen des Node-Files im Kommando CREATE-FILE angeben. Das CSS wird von der Quelldatei übernommen (*NONE entspricht EDF03IRV).
    - Wenn ein SAM-Node-File in eine SAM-Datei kopiert wird, wird das NETCSS nicht übertragen.
    - Wenn ein SAM-Node-File in ein SAM-Node-File kopiert wird, wird das NETCCS aus der Quelldatei übernommen.

- Der Administrator des File-Servers kann für Anwendungen, die auf unterschiedlichen Betriebssystemen ablaufen, den direkten Zugriff auf gemeinsame Dateien auf Net-Storage zulassen. Hierbei ist Folgendes zu beachten:
  - Anwendungen fremder Betriebssysteme (nicht BS2000) dürfen auf BS2000-Dateien auf Net-Storage nur lesend zugreifen. Auf Node-Files dürfen sie auch schreibend zugreifen.
  - Benutzer aus offenen Systemen erhalten durch ihre Benutzer- und Gruppennummer (uid, gid) Zugriff auf Node-Files. Für den gemeinsamen Zugriff auf Node-Files müssen die Benutzer- und Gruppennummer des BS2000-Benutzers und des Benutzers im offenen System miteinander abgestimmt werden. Unter NFSv4 ist dazu die Verwendung eines Verzeichnisdienstes (openLDAP oder Active Directory) notwendig.

  > **i** Zugriffsschutz in BS2000 (z.B. Zugriff nur Dateieigentümer, Kennwortschutz) hat nur dann eine Wirkung, wenn der Zugriff von BS2000 aus erfolgt. Für Zugriffe aus offenen Systemen hat er keine Wirkung.

3.4 Magnetbänder / Magnetbandkassetten

Magnetbänder und Magnetbandkassetten (MBK) sind private Datenträger, sie müssen mit einer PREMOUNT- oder MOUNT-Meldung angefordert werden. Im Gegensatz zu den Platten, gleich ob gemeinschaftliche oder private, kann ein Magnetband/eine Magnetbandkassette nicht von mehr als einem Auftrag gleichzeitig bearbeitet werden.

3.4.1 Magnetbänder

Magnetbandtypen werden über die Schreibdichte unterschieden, mit der die Daten aufgezeichnet werden (gemessen in bits per inch = bpi). Der Bandtyp, auf dem eine Datei aufgezeichnet wird, wird durch den DEVICE-Operanden des Makros FILE bzw. durch den Operanden DEVICE-TYPE in den Kommandos CREATE-FILE und IMPORT-FILE definiert.

3.4.2 Magnetbandkassetten

Für die Aufzeichnung von Daten auf Magnetbandkassetten (MBK) gilt die Norm DIN 66229.A; Vornorm ist dabei die für Magnetbandgeräte gültige Norm DIN 66029.

Die Datenübertragungsrate von/zur MBK wird durch Pufferung der Daten in der Steuerung des MBK-Gerätes stark erhöht. Allerdings kann der Benutzer festlegen, ob er beim Schreiben auf diese Pufferung verzichten will.

Generell verläuft die Verarbeitung von Dateien auf MBK wie die Verarbeitung von Dateien auf Magnetbändern. Programm-Inkompatibilitäten ergeben sich nur bei gepufferter Verarbeitung, wenn beim Schreiben der Daten ein nicht behebbarer Fehler auftritt und per Programmroutine Daten aus dem Puffer gerettet werden sollen oder wenn Fixpunkte erstellt werden sollen.
Bei ungepufferter Verarbeitung – die allerdings wesentlich langsamer ist – treten diese Inkompatibilitäten nicht auf.

Für weitere Hinweise zur Nutzung von MBK siehe Abschnitt „Verwendung von MBK-Systemen“.

3.5 Datenträger fremder Rechenzentren importieren

Privatplatten und Bänder können zwischen BS2000-Rechenzentren ausgetauscht werden. Eine Datei kann nur dann in ein fremdes System übertragen werden, wenn die Benutzerkennung des Eigentümers im fremden System eingetragen ist. Die Datei wird dann unter der Benutzerkennung des Eigentümers katalogisiert. Für Dateien auf Privatplatten entnimmt das DVS die Dateimerkmale dem F1-Kennsatz der Privatplatte, nicht den Angaben im Makro FILE bzw. im Kommando IMPORT-FILE oder einem bereits bestehenden Katalogeintrag.

Zur Übernahme in das fremde System gibt es folgende Möglichkeiten:

- Wenn mehrere Dateien von Privatplatten zu übernehmen sind, kann der Makro IMPORT bzw. das Kommando IMPORT-FILE verwendet werden, die auch teilqualifizierte Dateinamen verarbeiten.
- Wenn nur eine einzelne Datei zu übernehmen ist, lässt sich dies mit dem Makro FILE erreichen, der folgende Operanden enthält:

  ```
  pfadname
  STATE=FOREIGN
  VOLUME=vsn
  DEVICE=gerät
  ```

  Mit dem Kommando IMPORT-FILE kann auch eine einzelne Datei übernommen werden. Plattendateien können sowohl mit FILE- als auch mit IMPORT-Makro bearbeitet werden, Banddateien nur mit FILE-Makro. Mit dem Kommando IMPORT-FILE können sowohl Plattendateien als auch Banddateien bearbeitet werden.

3.6 Zusammenfassung

Die vorausgegangenen Abschnitte behandelten das Konzept der Datenträger im BS2000. Die folgende Tabelle bietet eine Übersicht über die Merkmale der Datenträger.

Merkmale der Datenträger	gemeinschaftliche Datenträger (Pubsets)		private Datenträger	
	Platte	Net-Storage [1)]	Platte	Band
Anzahl der Aufträge, die den Datenträger simultan nutzen können	beliebig	beliebig	beliebig (Ausnahme: exklusive Reservierung)	nur einer
Dateischutz	vom DVS vollständig gewährleistet	in BS2000 vom DVS vollständig gewährleistet[2)]	vom DVS vollständig gewährleistet	vom DVS vollständig gewährleistet
zulässige Verarbeitungs-Methoden	Zugriffsmethoden des DVS; Ein-/Ausgabe-Makros für Systemdateien (RDATA usw.)	Zugriffsmethoden des DVS; Ein-/Ausgabe-Makros für Systemdateien (RDATA usw.)	Zugriffsmethoden des DVS; Ein-/Ausgabe-Makros für Systemdateien (RDATA usw.)	Bandzugriffs-Methoden des DVS: SAM, UPAM und BTAM
Kennsätze			Standard-Kennsätze	Standard- und Nicht-Standard-Kennsätze
Archivnummer (VSN)	6-stellig, alphanumerisch; siehe "Gemeinschaftliche Platten (Pubsets)"	Standardname vom Puset abgeleitet (6-stellig, alphanumerisch); benutzerdefinierter Name wie Privatplatte	6-stellig, alphanumerisch; beginnt nicht mit „PUB“ und enthält keinen Punkt	6-stellig, alphanumerisch; beginnt nicht mit „PUB“ und enthält keinen Punkt
Datenträger-Anforderung	Pubsets brauchen vom Benutzer nicht angefordert zu werden	Net-Storage-Volume braucht nicht angefordert zu werden	Platten müssen vom Benutzer angefordert werden	Bänder müssen vom Benutzer angefordert werden
dynamische Speicherplatz-Verwaltung	ja	ja	ja	nein
Reservierung	nicht möglich	nicht möglich	erforderlich, sonst evtl. Auftrags-Abbruch, wenn der Datenträger nicht verfügbar ist	

Tabelle 4: Merkmale der Datenträger

1) Net-Storage erweitert den Speicherplatz auf gemeinschaftlichen Datenträgern, zählt aber selbst nicht zu den gemeinschaftlichen Datenträgern

2) Auf Seiten offener Systeme, die Zugriff zu Net-Storage-Dateien haben, können die BS2000-Schutzattribute nicht durchgesetzt werden. Standardmäßig sind Node-Files über Benutzer- und Gruppennummer (entspricht UID, GID) dem betreffenden Benutzer zugeordnet und es ist nur für diesen Benutzer das Lese- und Schreibrecht eingetragen. Jedoch können von UNIX-Seite die UNIX-Zugriffsrechte verändert werden (Details siehe Abschnitt „Randbedingungen“).

4 Dateien in BS2000

Datenmengen, die von Programmen verarbeitet werden sollen, werden zu Dateien zusammengefasst, aber auch die Programme selbst, Quellprogramme wie Lademodule, sind Dateien im Sinne von BS2000. Für die Verarbeitung von Daten muss der Benutzer jederzeit auf seine Dateien zugreifen können. Das DVS unterstützt ihn in allen Funktionen des Dateizugriffs, der Verwaltung und Wartung des Dateibestandes.

Die in einem System zu verarbeitenden Informationen (Daten) werden zu verschiedenen Einheiten zusammengefasst. Die kleinste adressierbare Einheit ist das Byte, die kleinste Übertragungseinheit ist der physikalische Block, der auf Platten eine feste Größe hat und als PAM-Seite bezeichnet wird (Abschnitt „Blockformate für Plattendateien“).

Logische Einheiten sind Zeichen, Feld, Satz, Datei und logischer Block oder Datenblock. Das Zeichen als kleinste logische Einheit entspricht dem Byte. Ein Feld kann ein oder mehrere Zeichen umfassen. Die Felder bilden den Datensatz, das Grundelement einer Datei. Die Datei schließlich ist eine Menge logisch zusammengehöriger Sätze.

Der Satz als logische Verarbeitungseinheit bleibt dem System (Ein-, Ausgabegeräte usw.) unbekannt. Der Datenblock stellt die Verbindung zwischen den Einheiten der Datenspeicherung und den Übertragungseinheiten /Verarbeitungseinheiten dar. Er kann als Übertragungseinheit z.B. einen oder mehrere Standardblöcke umfassen. Standardblöcke (PAM-Seiten) haben einen bestimmten Aufbau und eine bestimmte Länge, sie können für Platten- und Banddateien genutzt werden, Nicht-Standardblöcke nur für Banddateien (siehe Abschnitt „Blockformate für Plattendateien“ und Abschnitt „Blockformate für Banddateien“).

Der Datenblock kann einen oder mehrere Datensätze enthalten, die sich über eine oder mehrere PAM-Seiten erstrecken können.

4.1 Dateitypen

Generell sind drei Dateitypen zu unterscheiden: permanente Dateien, Arbeitsdateien und temporäre Dateien. Permanente und Arbeitsdateien sind nicht an den sie erzeugenden Auftrag gebunden. Temporäre Dateien sind an den Auftrag gebunden, in dem sie erzeugt wurden. Sie werden bei der LOGOFF-Behandlung automatisch gelöscht.

Auch SPOOLOUT-Dateien sind an den sie erzeugenden Auftrag gebunden. Diese werden jedoch nicht vom DVS verwaltet und sind im Handbuch „SPOOL“ [4] ausführlich beschrieben.

4.1.1 Permanente Dateien

Permanente Dateien können mit den Zugriffsmethoden SAM, ISAM, UPAM, FASTPAM, DIV oder BTAM erstellt werden. Sie liegen auf externen Datenträgern. Die Dateien bleiben auch nach Beendigung des Auftrags, in dem sie erstellt und katalogisiert wurden, bis zur expliziten Löschung mit dem Kommando DELETE-FILE erhalten. Dateinamen und Dateieigenschaften werden vom Benutzer festgelegt. Über im Katalogeintrag hinterlegte Dateimerkmale können auch andere Benutzer eine Zugriffsberechtigung für diese Dateien erhalten.

RFA-Zugriff ist möglich (Remote File Access, siehe Handbuch „RFA“ [6]). Permanente Dateien können auf gemeinschaftlichen Datenträgern, auf privaten Datenträgern (Privatplatten, Bändern) oder auf Net-Storage gespeichert werden. Auf gemeinschaftlichen Datenträgern (Pubsets) unterliegen sie der Pubspace-Kontrolle durch das System.

Permanente Dateien werden im Katalog unter einem „Pfadnamen“ geführt, der aus Katalogkennung (Catid), Benutzerkennung (Userid) und dem vom Benutzer vergebenen, vollqualifizierten Dateinamen besteht (siehe Abschnitt „Pfadname“).

4.1.2 Arbeitsdateien

Arbeitsdateien liegen in einem SM-Pubset auf einem eigens dafür vorgesehenen Volume-Set, der die Work-Eigenschaft besitzt (Work-Volume-Set). Arbeitsdateien werden entweder durch Angabe des Datei-Attributs WORK-FILE=*YES (z.B. im Kommando CREATE-FILE) oder über physikalische Allokierung (siehe "Physikalische Allokierung") erzeugt, z.B. mit dem Kommando

```
CREATE-FILE ...,VOLUME=<vsn einer Platte aus dem Work-Volume-Set>
               ,DEVICE-TYP=<zugehöriger Gerätetyp>).
```

Existiert in einem SM-Pubset kein Work-Volume-Set, kann keine Arbeitsdatei angelegt werden.
Arbeitsdateien können mit den Zugriffsmethoden SAM, ISAM, UPAM, FASTPAM und DIV verarbeitet werden.

Die Dateien bleiben auch nach Beendigung des Auftrags, in dem sie erstellt und katalogisiert wurden, bis zur expliziten Löschung mit dem Kommando DELETE-FILE erhalten. Dateiname und Dateieigenschaften werden vom Benutzer festgelegt. Über im Katalogeintrag hinterlegte Dateimerkmale können auch andere Benutzer eine Zugriffsberechtigung für diese Dateien erhalten.
RFA-Zugriff ist möglich (Remote File Access, siehe Handbuch „RFA“ [6]). Arbeitsdateien unterliegen nicht der Pubspace-Kontrolle durch das System.

Das Nutzungsmodell für Arbeitsdateien besteht darin, dass sie ausschließlich auf solchen Work-Volume-Sets angelegt werden, die der Systembetreuer den Benutzern zeitweise zur Verfügung stellt. Über die Dauer der Verfügbarkeit eines solchen Work-Volume-Sets müssen sich Systembetreuer und Benutzer absprechen. Nach Ablauf der festgelegten Frist hat der Systembetreuer das Recht, alle auf dem Work-Volume-Set befindlichen Dateien zu löschen und den Work-Volume-Set selber zu exportieren.

Bezüglich des Pfadnamens unterscheiden sich Arbeitsdateien nicht von permanenten Dateien.

4.1.3 Temporäre Dateien

Temporäre Dateien sind auftragsbezogen. Es können weder andere Aufträge derselben Benutzerkennung auf sie zugreifen, noch andere Benutzer eine Zugriffsberechtigung erhalten. Bei Auftragsende werden sie im Rahmen der LOGOFF-Behandlung gelöscht, sofern sie nicht schon vorher vom Eigentümer explizit gelöscht wurden. Steht zum LOGOFF-Zeitpunkt noch ein SPOOLOUT-Auftrag für eine temporäre Datei an, wird sie erst nach Beendigung des SPOOLOUT-Auftrags gelöscht. Kommt es wegen fehlerhafter Systembeendigung zum Auftragsabbruch, bleiben die temporären Dateien wegen der fehlenden LOGOFF-Behandlung katalogisiert, bis die Systembetreuung den Pubset erneut importiert.

Temporäre Dateien können mit den Zugriffsmethoden ISAM, SAM, UPAM, FASTPAM, DIV oder BTAM verarbeitet werden. Sie werden standardmäßig im Benutzerbereich des Systems angelegt und – wo zulässig – vom DVS wie permanente Dateien desselben Typs behandelt. Sie können zwar auf Bändern, nicht jedoch auf Privatplatten angelegt werden.

Der nichtprivilegierte Benutzer kann temporäre Dateien nur auf dem Default-Pubset seiner Benutzerkennung anlegen.

Der Benutzer kann eine temporäre Datei erzeugen, indem er dem Dateinamen ein Sonderzeichen voranstellt. Dieses Sonderzeichen (# oder @) wird durch den Systemparameter TEMPFILE bei der Systemeinleitung festgelegt und kann mit dem Kommando SHOW-SYSTEM-PARAMETER bzw. dem Makro NSIOPT (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [2]) abgefragt werden.
Der vom Benutzer festgelegte Name für eine temporäre Datei lautet dann z.B. „#KUNDEN-LISTE“. Unter diesem Namen kann er die temporäre Datei bearbeiten (öffnen, schließen, löschen, usw.).

Temporäre Dateien werden im Dateikatalog unter einem internen Dateinamen geführt, der folgende Form aufweist:
S.<sysid>.<tsn>.<dateiname>

<sysid>	bezeichnet die Identifikation des Systems im Verbund (dreistellige Ziffer)
<tsn>	bezeichnet die aktuelle Auftragsnummer TSN (Task Sequence Number)

Der interne Dateiname einer temporären Datei kann z.B. folgendermaßen aussehen: S.100.1QXR.TEMP.DATEI

Bei verschiedenen Makros sowie in Systemmeldungen wird der interne Dateiname ausgegeben. Durch den Aufbau des Dateinamens ist gewährleistet, dass gleichbenannte temporäre Dateien verschiedener gleichzeitig aktiver Aufträge eines Benutzers nicht kollidieren.

Wird ein Dateiname vergeben, der dem Aufbau eines temporären Dateinamens entspricht (Dateiname beginnt mit „S.“, gefolgt von einer dreistelligen Zahl, einem Punkt und anschließend einer vierstelligen Buchstaben-/Ziffernkombination), wird dieser vom System zurückgewiesen.

Legt der Benutzer eine Datei explizit mit einem Dateinamen an, der über die genannten Merkmale (S.<sysid>.<tsn>) verfügt, wird die Datei als temporäre Datei angelegt. Zur Bearbeitung dieser Datei kann bzw. soll der Dateiname mit vorangestelltem Sonderzeichen (s.o.) angeben werden. Nach Beendigung der Task ist die Datei verloren.

> **i** Die Verwendung der „internen“ Dateinamen und deren Verankerung in Programmen/Prozeduren macht eine Portabilität dieser Produkte in andere Betriebssystemversionen unmöglich. Der interne Name einer temporären Datei ist kein Bestandteil der Benutzerschnittstelle und kann jederzeit geändert werden. Er sollte daher nicht in Programmen/Prozeduren verwendet werden.

Werden Banddateien als temporäre Dateien (mit Präfix im Dateinamen) eingerichtet, wird weder das Präfix noch die Zeichenfolge „S.<sysid>.<tsn>“ in die Kennsätze HDR1 oder HDR3 übernommen (nur Dateiname ohne Präfix). Nur im Katalogeintrag ist die Datei als temporäre Datei gekennzeichnet. Wird eine solche Datei mit OPEN INPUT oder OPEN INOUT (siehe Kapitel „OPEN-Verarbeitung“) geöffnet, überprüft das DVS bei HDR1-/HDR3-Verarbeitung diese Zeichenfolge nicht.

Temporäre Dateigenerationsgruppen sind nicht erlaubt.

Das Umkatalogisieren einer Arbeitsdatei in eine temporäre Datei oder umgekehrt ist nicht möglich.

In einem SM-Pubset wird das Umbenennen von temporär zu permanent und umgekehrt abgewiesen, wenn eine gleichzeitige Änderung der Dateiattribute eine Umallokierung auf einen anderen Volume-Set (S0-Migration) erfordert.

EAM-Dateien

Der SYSEAM-Bereich ist ein unter der Benutzerkennung TSOS geführter Bereich auf gemeinschaftlichen Datenträgern. Die hier vom Benutzer oder vom System angelegten Dateien sind temporär und können mit der Zugriffsmethode EAM verarbeitet werden (siehe Kapitel „EAM – Evanescent Access Method“ (EAM - Evanescent Access Method)).
Diese EAM-Dateien unterliegen anderen Konventionen als die übrigen temporären Dateien, z.B. in der Namensgebung: beim ersten Eröffnen einer EAM-Datei weist das System der Datei eine 2-Byte-Binärzahl (1 <= name <= 14000) als Dateinamen zu, mit dem die Datei angesprochen werden kann. Im Unterschied zu anderen temporären Dateien werden EAM-Dateien nicht im Benutzerkatalog geführt.

Eine besondere EAM-Datei ist die vom System angelegte Bindemoduldatei. Sie kann als *-Datei bzw. *OMF-Datei angesprochen werden (z.B. wird mit dem Makro ERASE * oder dem Kommando DELETE-SYSTEM-FILE *OMF die Bindemoduldatei gelöscht). Auch der EAM-Makro bietet Zugriff auf die Bindemoduldatei an.

Logische Systemdateien

Das BS2000 benutzt Systemdateien zur Anweisungs- und Dateneingabe an das Betriebssystem bzw. zur Ausgabe von Daten oder Meldungen durch das Betriebssystem. Diese vom System vorgegebenen Ein- und Ausgabebereiche, die sog. logischen Systemdateien, sind temporäre Dateien, da sie auftragsabhängig sind.

Das Arbeiten mit logischen Systemdateien hat keine Auswirkungen auf die Pubspace-Belegung des Benutzers.

Die Gesamtheit der Systemdateien, die einem Auftrag zur Verfügung stehen, wird als SYS-FILE-Umgebung bezeichnet. Sie spielt zum Beispiel eine Rolle bei der Fixpunktschreibung (siehe Makro WRCPT, Handbuch „Makroaufrufe an den Ablaufteil“ [2]).

Systemdateien können als Ein- oder Ausgabewege des Systems genutzt werden, z.B. mit den Makroaufrufen WRLST und RDATA (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [2]).

Das DVS des BS2000 unterstützt nur die Verarbeitung permanenter und temporärer Benutzerdateien sowie EAM-Dateien. Für die Verarbeitung logischer Systemdateien wird daher auf die Handbücher „Kommandos“ [3] und „Makroaufrufe an den Ablaufteil“ [2] verwiesen.

4.2 Dateinamen

Jede im BS2000 katalogisierte Datei ist durch einen sog. Pfadnamen eindeutig identifizierbar. Er setzt sich zusammen aus einem vom Eigentümer vergebenen Dateinamen und einem vom System vergebenen „Vorspann“, der dem Zugriffspfad im Pubset entspricht.

Zu unterscheiden sind folgende Begriffe: Pfadname, voll- und teilqualifizierter Dateiname, interner Dateiname. Für den Dateinamen temporärer Dateien kann auch die Bezeichnung „Tempfile-Name“ verwendet werden.

4.2.1 Pfadname

Im Dateikatalog ist der Dateiname und – an anderer Stelle – die Benutzerkennung eingetragen. Der Pfadname ist der Name einer BS2000-Datei einschließlich Katalog- und Benutzerkennung. Er bezeichnet eindeutig eine permanente oder temporäre Datei, eine Dateigeneration oder eine Dateigenerationsgruppe.

Aufbau

pfadname = :catid:$userid.dateiname

Die Gesamtlänge des Pfadnamens beträgt maximal 54 Zeichen.

catid

Katalogkennung; Kennzeichen des Pubsets, in dem die Datei katalogisiert ist; Länge: 1 bis 4 Zeichen;

Default-Catid = Standard-Katalogkennung:
Sie ist die der Benutzerkennung im Benutzerkatalog zugewiesene Katalogkennung, d.h. die Kennung des Pubsets, auf dem die Dateien des Benutzers standardmäßig angelegt werden.

Die Katalogkennung muss immer in Doppelpunkte eingeschlossen werden (*:catid:*)

userid

Benutzerkennung; Länge: max. 8 Zeichen

Default-Userid = Standard-Benutzerkennung:
Sie ist im Normalfall die Benutzerkennung des laufenden Auftrags, d.h. die des Kommandos SET-LOGON-PARAMETERS; beim „Secondary Read“ kann eine Default-Userid auf Systemebene genutzt werden, unter der Dateien katalogisiert sind, die allen Benutzern zugänglich sein sollen (siehe Abschnitt „System-Standardkennung (SystemDefault Userid)“ und "Zugriff über die System-Standardkennung").

Die Angabe der Benutzerkennung bei Bezugnahme auf Dateien erfolgt immer mit vorangestelltem $-Zeichen und abschließendem Punkt ($*userid.*)

dateiname

Vom Benutzer festgelegter vollqualifizierter Dateiname;
Seine Länge ist von der Dateiart abhängig:

- für permanente Dateien maximal 41 Zeichen
- für temporäre Benutzerdateien maximal 31 Zeichen (einschließlich Präfix)
- für Dateigenerationsgruppen maximal 34 Zeichen

Die maximal mögliche Länge verringert sich, wenn bei einer mehrstelligen Katalogkennung die Summe aus Länge der Katalogkennung (einschließlich Doppelpunkte) und Länge der Benutzerkennung (einschließlich $-Zeichen und Punkt) die Länge von 13 Zeichen überschreitet (**:***catid:$userid.*).

4.2.2 Vollqualifizierter Dateiname

Ein vollqualifizierter Dateiname ist der vom Benutzer vergebene Dateiname, der mit Katalog- und Benutzerkennung verknüpft wird. Ein vollqualifizierter Dateiname kann durch Punkte in Teilnamen gegliedert sein.

Aufbau

Vollqualifizierter Dateiname = $name_1$[.$name_2$[. ...]][({*absgen / version})]

Die Gesamtlänge ist von der Dateiart abhängig:

- für permanente Dateien maximal 41 Zeichen
- für temporäre Dateien maximal 31 Zeichen
- für Dateigenerationsgruppen maximal 34 Zeichen
- für Dateigenerationen maximale Länge des Namens einer Dateigenerationsgruppe (s. o.) plus 7 Zeichen für die absolute Generationsnummer

$name_1$

Teilname-1 (siehe „Regeln für die Bildung von Dateinamen“)

$name_2$

Teilname-2 (siehe „Regeln für die Bildung von Dateinamen“)

absgen

Absolute Generationsnummer;
Größenordnung: 1 <= absgen <= 9999;
Die Generationsnummer identifiziert Dateigenerationen innerhalb einer Dateigenerationsgruppe. Dateigenerationen können auch über relative Generationsnummern angesprochen werden; relative Generationsnummern werden jedoch nicht in den Katalogeintrag aufgenommen (siehe Abschnitt „Generationsnummer“).

Die absolute Generationsnummer muss immer mit einem führenden Stern und in runde Klammern eingeschlossen angegeben werden (*(*absgen)*).

version

Versionsbezeichung für Banddateien; einfacher Name aus einem oder mehreren Zeichen (Buchstaben, Ziffern oder Sonderzeichen)

Katalogisierte Banddateien müssen im Allgemeinen eindeutige Dateinamen haben. Mit der freiwählbaren Versionsbezeichnung können verschiedene Dateien mit ansonsten gleichem Datei-/Pfadnamen katalogisiert werden.

Die Versionsbezeichnung wird nur in den Dateikatalog, nicht aber in die Dateikennsätze übernommen. Sie ist somit nur ein Hilfsmittel, das den Katalogeintrag eindeutig identifiziert und die implizite Zuordnung von Datei und Datenträgern ermöglicht.

Die Version muss immer in runde Klammern eingeschlossen werden (*(version)*).
Das DVS erkennt eine Versionsbezeichnung an der ersten Klammer und dem darauffolgenden Zeichen, das nicht ein Plus, Minus oder ein Stern ist (+ oder – oder * in der Klammer kennzeichnen Dateigenerationen).
Das DVS ignoriert beim Übertragen des Dateinamens in die Kennsätze alle Zeichen ab der Klammer. Beim Lesen von Kennsätzen werden die Dateinamen in Kennsatz und Katalogeintrag nur bis zum Auftreten der ersten Klammer verglichen; alle weiteren Zeichen ab der Klammer werden ignoriert.

Regeln für die Bildung von Dateinamen

Dem Benutzer steht folgender Zeichenvorrat zur Verfügung:

- alle Buchstaben (A ... Z)
- alle Ziffern (0 ... 9)
- die Sonderzeichen #, @, $ sowie Bindestrich (-) und Punkt (.).

Der vollqualifizierte Dateiname muss mindestens einen Buchstaben enthalten, der an beliebiger Stelle innerhalb der Gesamtzeichenfolge stehen kann.

Der Punkt gliedert einen vollqualifizierten Dateinamen in Teilnamen ($name_1$.$name_2$). Er kann also nicht erstes oder letztes Zeichen eines Teilnamens sein.

Der Bindestrich darf nicht erstes oder letztes Zeichen eines Teilnamens sein.

Das $-Zeichen hat als Kennzeichnung für die Benutzerkennung eine Sonderfunktion und darf deshalb nicht als erstes Zeichen eines Dateinamens angegeben werden.

Die Zeichen @ und # haben eine Sonderfunktion als Präfix zur Kennzeichnung temporärer Dateien. Sie sollten daher diese Zeichen nicht als erstes Zeichen eines Dateinamens verwenden, auch wenn die TEMPFILE-Funktion (d.h. die Zulässigkeit temporärer Dateien) im System nicht aktiv ist.

Es wird empfohlen, bei der Vergabe von Datei- und Gruppennamen deren Länge stets auf den Maximalwert für vierstellige Katalog- und achtstellige Benutzerkennungen zu begrenzen. Damit wird sichergestellt, dass der Dateibestand problemlos auf ein anderes Pubset bzw. eine andere Benutzerkennung übertragen werden kann (z.B. Wechsel der Benutzer von Pubset „A“ auf Pubset „AB01“).

Beispiel

```
/show-file-attributes # ————————————————————————————————————  (1)
         3 :2OSG:$USERXYZ.S.152.500H.PROTOKOLL
         3 :2OSG:$USERXYZ.S.152.500H.VOLL.QUALIF.TEMPFILE.NAME
:2OSG: PUBLIC:      2 FILES RES=         6 FRE=         6 REL=        6 PAGES
/show-file-attributes  generations=yes ———————————————————————  (2)
         0 :G:$USERXYZ.ABCDEFGHIJKLMNOPQRSTUVWXYZ.1234567890 ——————  (3)
         3 :G:$USERXYZ.ABDEF.134.$-@#
         3 :G:$USERXYZ.DATEINAME
         0 :G:$USERXYZ.DAT.GENGROUP (FGG) ———————————————————————  (4)
         0 :G:$USERXYZ.DAT.GENGROUP(*0001)
         0 :G:$USERXYZ.DAT.GENGROUP(*0002)
         3 :G:$USERXYZ.DO.CMH.ASS ———————————————————————————————  (5)
        12 :G:$USERXYZ.DVS.TAB2-2
        30 :G:$USERXYZ.PROTO.DATEINAMEN
         3 :G:$USERXYZ.VOLL.QUALIF.DATEI.NAME
         3 :G:$USERXYZ.1234567890X0987654321
:2OSG: PUBLIC:     11 FILES RES=        57 FRE=        47 REL=       36 PAGES
```

(1) Das Kommando SHOW-FILE-ATTRIBUTES listet die unter der Aufrufer-Benutzerkennung ($USERXYZ) auf dem Standard-Pubset (2OSG) vom laufenden Auftrag (TSN=500H) erzeugten temporären Benutzerdateien (Präfix: #) auf.

(2) Das Kommando SHOW-FILE-ATTRIBUTES listet die unter der Aufrufer-Benutzerkennung ($USERXYZ) auf dem Standard-Pubset (2OSG) katalogisierten permanenten Dateien, Dateigenerationsgruppen und Dateigenerationen auf.

(3) 3 Beispiele für den erlaubten Zeichenvorrat bei der Bildung von Dateinamen.

(4) Beispiel für eine Dateigenerationsgruppe mit 2 Dateigenerationen.

(5) 5 Beispiele für den erlaubten Zeichenvorrat bei der Bildung von Dateinamen.

4.2.3 Teilqualifizierter Dateiname

Ein Dateiname heißt teilqualifiziert, wenn er mit einem Punkt endet. Durch den Aufbau eines Dateinamens aus mehreren Teilnamen ist es möglich, Zuordnungen oder Funktionen im Dateinamen abzubilden. Über einen teilqualifizierten Dateinamen können mehrere Dateien gleichzeitig angesprochen werden. Daher können teilqualifizierte Dateinamen nicht beim Einrichten einer Datei verwendet werden, sondern nur bei Zugriffen auf bereits bestehende Dateien (löschen, importieren, informieren usw.).

Aufbau

Teilqualifizierter Dateiname = $name_1$.[$name_2$.]...

$name_1$= Teilname-1

$name_2$ =Teilname-2

Das letzte Zeichen eines teilqualifizierten Dateinamens ist immer der Punkt.

Ein teilqualifizierter Dateiname bezieht sich immer auf eine oder mehrere existente/katalogisierte Dateien. Die Regeln für die Angabe von $name_1$, $name_2$ sind daher dem Abschnitt„Vollqualifizierter Dateiname“ zu entnehmen.

Beispiel

Zu erstellten Quellprogrammen in verschiedenen Programmiersprachen (z.B. Assembler, und FORTRAN) werden Übersetzungs- und Ablaufprotokolle während der Programm-Testphase und schließlich Lademodule erzeugt. Mögliche Dateinamen wären:

Quellprogramme	Übersetzungsprotokolle	Ablaufprotokolle	Lademodule
`Q.ASS.PROG1.V1`	`ASSLST.PROG1.V1`	`PROTO.PROG1.V1.1`	`L.PROG1.V3`
`Q.ASS.PROG1.V2` `Q.ASS.PROG1.V2A`	`ASSLST.PROG1.V2`	`PROTO.PROG1.V1.2` `PROTO.PROG1.V2.1`	`L.PROG1.V2` `L.PROG3.V3`
`Q.ASS.PROG1.V3`			
`Q.ASS.PROG2.V1` `Q.FOR.PROG4.V1`			
`Q.FOR.PROG4.V2`			
`Q.FOR.PROG5.V3`			

Mit dem Makroaufruf `ERASE ASSLST.` bzw dem Kommandoaufruf `/DELETE-FILE ASSLST.` können alle bei der Übersetzung erzeugten Dateien mit $name_1$=ASSLST gelöscht werden. Mit ERASE bzw. `/DELETE-FILE PROTO.` alle so benannten Protokolle, usw.
Die Angabe „PROTO.PROG1.V1.“ listet alle die erste Version von PROG1 betreffenden Ablaufprotokolle.
Der Makroaufruf `FSTAT Q.` bzw. der Kommandoaufruf `/SHOW-FILE-ATTRIBUTES Q.` listet die Namen aller mit Q. bezeichneten Quellprogramme auf, `FSTAT Q.ASS.` bzw. `/SHOW-FILE-ATTRIBUTES Q.ASS.` alle in Assembler geschriebenen Quellprogramme, usw.

4.2.4 System-Standardkennung (System Default Userid)

Die Systembetreuung legt bei der Systemeinleitung mit dem Systemparameter DEFLUID eine systemweit gültige Standard-Benutzerkennung (in der Regel $TSOS) fest, unter der die nicht-privilegierten Software-Produkte katalogisiert sind, die allen Benutzern zur Verfügung stehen, z.B. Dienstprogramme, Dateiaufbereiter, Übersetzer usw.

Auf diese Dateien kann ohne Angabe einer Benutzerkennung zugegriffen werden, wenn dem Dateinamen die Zeichenfolge „$." oder das $-Zeichen vorangestellt wurde.
Ist der Dateiname durch Punkte in Teilnamen gegliedert, muss ihm die Zeichenfolge „$." vorangestellt werden. Ist der Dateiname nicht gegliedert, genügt es, das $-Zeichen voranzustellen.

Die System-Standardkennung spielt auch eine Rolle beim sog. „Zweiten Lesen" (siehe Abschnitt „Zugriff über die System-Standardkennung").

Beispiel

```
Dateiaufbereiter: $.EDT, $EDT
Übersetzer: $ASSEMB
Dienstprogramme: $SORT, $PERCON, $DPAGE, $TSOSLNK, $LMS, $.ARCHIVE
```

4.3 Dateien größer 32 GB

Dem stetigen Wachstum der Plattenspeicherkapazität und von online bereitzuhaltenden Datenbeständen wird im B2000 durch die Erweiterung der früher verfügbaren Platten- und Dateigrößen von ca. 32 GB Rechnung getragen. Es gelten folgende Grenzen:

- Die maximale Kapazität eines Pubsets oder Volume-Sets beträgt ca. 4TB.
- Die maximale Kapazität einer einzelnen Platte beträgt ca. 2 TB.
- Die maximale Dateigröße beträgt ca. 4 TB (entspricht der maximalen Größe eines Pubsets oder Volume-Sets abzüglich SVL, F5-Label und Systemdateien).

Dies bedeutet, dass innerhalb des Betriebssystems konsequent 4-Byte-Blocknummern und 4-Byte-Zähler für Datei- und Plattengrößen verwendet werden müssen.

Die Umstellung von 3-Byte- auf 4-Byte-Felder hat Auswirkungen auf alle Komponenten, Produkte und Anwendungen, die mit diesen Feldern direkt oder indirekt arbeiten. Die TPR-Schnittstellen wurden entsprechend angepasst, aber nicht alle TU-Benutzerschnittstellen können diese großen Dateien kompatibel unterstützen.

Voraussetzungen

Es gibt zwei Pubset-Typen für große Objekte (große Dateien und Volumes):

- LARGE-OBJECTS-Pubsets ohne große Dateien:

  Dieser Typ erlaubt große Volumes, beschränkt aber die zulässige Dateigröße auf 32 GB. Dies bedeutet, dass der Pubset Volumes >= 32 GB enthalten darf, nicht aber notwendigerweise aktuell derartige Volumes enthält.

  Diese Pubsets verhalten sich aus Anwendersicht (fast) wie konventionelle Pubsets.
- LARGE-OBJECTS-Pubsets mit großen Dateien

  Bei diesem Typ sind große Volumes und große Dateien zulässig, nicht aber notwendigerweise vorhanden.

  Er ermöglicht die Abschottung von großen Dateien gegen Programme, die unverträgliche Schnittstellen nutzen, und unterstützt die stufenweise Einführung von großen Volumes und – in einem zweiten Schritt – großen Dateien.

Große Objekte werden sowohl auf SF- als auch auf SM-Pubsets unterstützt. Dazu wurden die zwei folgenden Attribute eingeführt:

LARGE_OBJECTS	Attribut zur Unterstützung von Volumes >= 32 GB
LARGE_FILES_ALLOWED	Attribut steuert, ob auf dem Pubset auch große Dateien (Dateien >= 32 GB) unterstützt werden

Analog zu Pubsets wurde für Volumes ein Attribut eingeführt: LARGE_VOLUME.
Das Attribut LARGE_VOLUME charakterisiert ein logisches Volume und zeigt an, dass das Volume eine Bruttokapazität >= 32 GB hat. Wie die Eigenschaft LARGE_OBJECTS für Pubsets ist die Eigenschaft LARGE_VOLUME für Volumes dauerhaft und wird im SVL abgelegt.

3-Byte- und 4-Byte-Felder

Zentral für die Aufhebung der 32-GB-Grenze für die Volume- und Dateigröße ist die Einführung von 4-Byte-Feldern für folgende im Katalogeintrag abgelegte Daten:

- FILE-SIZE – der für die Datei allokierte Speicherplatz

- HIGHEST-USED-PAGE – der davon aktuell durch Daten belegte Speicherplatz
- LHP (Logical halfpage number) und PHP (physical halfpage number) der einzelnen Extents in der Extent-Liste, die den logischen Halbseiten „physikalische“ Halbseiten von Volumes zuordnet.

Blocknummern und Blockzähler sind an verschiedenen Benutzerschnittstellen des BS2000 sichtbar. Während alle neueren Ausprägungen dieser Schnittstellen konsequent 4-Byte-Felder verwenden, werden bei manchen älteren Schnittstellenversionen 3-Byte-Felder verwendet. Sind Dateien >= 32 GB vorhanden, ist mit Kompatibilitätsproblemen zu rechnen; ebenso, wenn „nur“ Volumes >= 32 GB vorhanden sind.

Mit Einführung von 4-Byte-LHPs und 4-Byte-PHPs existiert auch für die Extent-Liste ein zweites (zusätzliches) Format. Um so weit wie möglich Abwärtskompatibilität sicherzustellen, unterstützen die aktuellen BS2000-Versionen beide Formate der Extent-Liste:

- Grundsätzlich wird das „alte“ Format mit 3-Byte-Blocknummern verwendet.
- Nur im Fall von großen Dateien oder von Dateien, für deren Adressierung PHPs größer X'FFFFFF' benötigt werden, wird das neue Format mit 4-Byte-Blocknummern verwendet.

Extent-Listen enthalten also entweder Extents mit 3-Byte-Blocknummern oder Extents mit 4-Byte-Blocknummern.

Einschränkungen für große Dateien

Im Bereich der nichtprivilegierten Anwendungen sind folgende Einschränkungen für große Dateien zu beachten:

- Die Unterstützung von Dateien >= 32 GB ist nur auf Pubsets mit LARGE_OBJECTS=TRUE, LARGE_FILES_ALLOWED=TRUE möglich.
- Die Unterstützung von Dateien >= 32 GB ist nur für Nicht-Pamkey-Dateien möglich: Dateien mit BLKCTRL=PAMKEY werden nicht unterstützt, da im Systemteil des Pamkey die logische Seitennummer als 3-Byte-Feld hinterlegt ist.
- EAM-Dateien >= 32 GB sind nicht möglich, da die Behälterdatei SYSEAM keine große Datei sein kann.

Übersicht über die 32-GB-relevanten DVS-Schnittstellen zur Datei-Verwaltung

Schnittstelle	Änderung
nicht-privilegierte Kommandos	
ADD-FILE-LINK	neuer Operand EXCEED-32GB für Angabe der erlaubten Dateigröße (größer oder kleiner 32 GB)
SHOW-FILE-LINK	Ausgabe des Dateiattributes „große Datei“
SHOW-FILE-ATTRIBUTES	erweiterte Ausgabestellen verschiedener Ausgabefelder
Makros	
FCB	neuer Operand LARGE_FILE für große Dateien
FILE	neuer Operand EXC32GB für große Dateien
FSTAT	Prüfungs- und Umstellungsaufwand bei VERSION=0/1

OPEN	Semantikproblem beachten
RDTFT	Ausgabe des Dateiattributes „große Datei“
DIV	neuer Operand LARGE_FILE für große Dateien; vergrößerter Wertebereich für die Operanden BLOCK und SPAN
FPAMACC	vergrößerter Wertebereich für den Operanden BLOCK
FPAMSRV	neuer Operand LARGE_FILE für große Dateien

Tabelle 5: Übersicht über die 32-GB-relevanten DVS-Schnittstellen zur Datei-Verwaltung

Ablauffähigkeit von Programmen in Konfigurationen mit Dateien größer 32 GB

Es kann nicht davon ausgegangen werden, dass alle Programme für den Zugriff auf große Objekte vorbereitet sind, d.h. mit 4 Byte breiten Blocknummern und Blockzählern zurecht kommen, wobei Benutzerprogrammen nur Schnittstellen für Zugriff auf und Bearbeitung von Dateien und deren Metadaten zur Verfügung stehen. Deshalb beschränkt sich die folgende Betrachtung auf große Dateien. Dabei lässt sich das Verhalten von Programmen wie folgt klassifizieren:

Klasse A Ein Programm ist in der Lage, große Dateien ohne Einschränkung zu verarbeiten. Dieses Verhalten wird als LARGE-FILES-fähig bezeichnet.

Klasse B: Ein Programm ist zwar nicht auf die Bearbeitung großer Dateien und/oder ihrer Metadaten vorbereitet, ist aber in der Lage, entsprechende Zugriffe, die als fehlerhaft betrachtet werden müssen, definiert abzuweisen oder es erfolgen im Programm keine Zugriffe auf Dateien und ihre Metadaten. Dieses Verhalten wird als LARGE-FILES-kompatibel bezeichnet.

Klasse C: Ein Programm ist nicht auf die Bearbeitung großer Dateien vorbereitet und ist auch nicht in der Lage, entsprechende Zugriffe definiert abzuweisen. Dieses Verhalten wird als LARGE-FILES-inkompatibel bezeichnet.

Für Konfigurationen, die große Dateien beinhalten, müssen LARGE-FILES-kompatible oder -fähige Programme vorausgesetzt werden. Dabei ist als Regelfall LARGE-FILES-Kompatibität zu sehen. Das Wachstum über 32 GB hinaus dürfte sich vorerst auf relativ wenige Dateien beschränken; nur Programme, die auf diese zugreifen, müssen LARGE-FILES-fähig sein.

Eine Zusammenstellung aller Aspekte des Einsatzes von großen Volumes und Dateien finden Sie im Handbuch „Dateien und Volumes größer 32 GB“ [18].

4.4 Dateiattribute für Ausgabe von Dateien

Das Dateiattribut CODED-CHARACTER-SET identifiziert den Zeichensatz der Datei über den CCS-Namen. Dieser Zeichensatz legt für die Zeichen des Dateiinhalts die Darstellung auf Ausgabemedien (zum Beispiel Bildschirm, Drucker) und die Sortierreihenfolge fest.

Das Dateiattribut CODED-CHARACTER-SET wird z.B. ausgewertet von den Kommandos COMPARE-DISK-FILES, MAIL-FILE, PRINT-DOCUMENT, SHOW-FILE sowie den Software-Produkten EDT und openFT.

Für eine Benutzerkennung kann ein Benutzer-Standard-Zeichensatz eingetragen werden. Bei der Festlegung des Dateiattributs CODED-CHARACTER-SET kann vereinbart werden, dass der Benutzer-Standard-Zeichensatz aus dem Benutzereintrag übernommen werden soll.

Im Systemparameter HOSTCODE kann ein System-Standard-Zeichensatz eingetragen sein. Bei der Festlegung eines Benutzer-Standard-Zeichensatzes kann vereinbart werden, dass der System-Standard-Zeichensatz übernommen werden soll.

Näheres finden Sie im Handbuch „XHCS“ [21].

Für Node-Files existiert zusätzlich das Dateiattribut NET-CODED-CHAR-SET, das den Zeichensatz identifiziert, mit dem die Datei auf Net-Storage gespeichert ist. Dieser ist für die Verarbeitung von SAM-Node-Files relevant (siehe Abschnitt „Zeichensatz für Node-Files“).

Im Systemparameter NETCODE ist der Standard für Zeichensätze von Node-Files auf Net-Storage definiert. Dieser Wert kann als NET-CODED-CHAR-SET in den Benutzer-Eintrag übernommen werden (siehe Kommando ADD-USER bzw. MODIFY-USER-ATTRIBUTES).

4.5 Dateivergleich von Plattendateien

Das Kommando COMPARE-DISK-FILES bzw. der Makro COMPFIL vergleicht zwei Plattendateien blockweise (UPAM) oder satzweise (SAM, ISAM) und informiert den Benutzer über das Vergleichsergebnis.

Es können auch temporäre Dateien oder Arbeitsdateien verglichen werden. Die Dateien können auf gemeinschaftlichen Datenträgern, Net-Storage oder Privatplatten liegen.

Dateien mit folgenden Eigenschaften können **nicht** verglichen werden:

- leere Dateien
- geöffnete Dateien
- gesperrte Dateien (z.B. SECURE-Lock)
- Kennzeichen REPAIR-NEEDED gesetzt
- Kennzeichen NO-DMS-ACCESS gesetzt

Dateigenerationsgruppen als Ganzes können nicht verglichen werden, wohl aber einzelne Dateigenerationen.

PLAM-Bibliotheken können blockweise verglichen werden. Ein Vergleich der enthaltenen Elemente ist nicht möglich.

5 Datei- und Katalogverwaltung

5.1 Dateikatalog

Dateien im BS2000 werden i.A. unter der Benutzerkennung des Auftrags katalogisiert, in dem sie erstellt werden. (Ausnahmen: Bei Miteigentümerschaft, siehe "Festlegung einer Miteigentümerschaft (Co-Owner)", oder mit TSOS-Privileg kann eine Datei auch unter fremder Benutzerkennung katalogisiert werden.)
Der Katalogeintrag beschreibt Dateieigenschaften und Datenträger; nur katalogisierte Dateien sind dem System zugänglich (Ausnahme: FOREIGN-Dateien auf Band).

5.1.1 Katalogeintrag erstellen

Das Erstellen eines Katalogeintrags hängt nicht vom Erstellen einer Datei (mit Speicherplatzbelegung) ab. Katalogeinträge können mit Makros und mit Kommandos erzeugt werden.

Makros

Katalogeinträge können mit den Makros CATAL und FILE erzeugt werden. CATAL und FILE haben dabei unterschiedliche Funktionen:

- CATAL erzeugt einen Eintrag im Benutzerkatalog mit Berücksichtigung der vom Datei-Eigentümer vergebenen Schutzmerkmale, jedoch ohne Speicherplatzreservierung.
- FILE erstellt einen Katalogeintrag und weist der Datei Speicherplatz zu (Ausnahme: Dateien auf Net-Storage-Volumes), übernimmt für die Dateischutzmerkmale jedoch nur Standardwerte. Diese Standardwerte können anschließend mit CATAL verändert werden. Die übrigen Dateieigenschaften, die in den Katalogeintrag übernommen werden (Zugriffsmethode bei Dateierstellung, Block- und Satzlänge usw.), werden beim ersten Öffnen der Datei eingetragen. Werte zur logischen Dateigröße (= die höchste belegte Seitennummer) u.Ä. werden erst beim Schließen der Datei übernommen.

  Um ein Node-File zu erstellen, muss im Makro FILE explizit NFTYPE=NODE-FILE angegeben werden. Im anderen Fall wird die Net-Storage-Datei als BS2000-Datei angelegt.

Kommandos

Katalogeinträge können mit folgenden Kommandos erzeugt werden:

- CREATE-FILE für Dateien
- CREATE-FILE-GENERATION für Generationen
- CREATE-FILE-GROUP für Dateigenerationsgruppen

Die Kommandos erzeugen einen Eintrag im Dateikatalog mit Berücksichtigung der vom Aufrufer vergebenen Schutzmerkmale sowie der bei CREATE-FILE und CREATE-FILE-GENERATION möglichen Einstellung der Speicherplatzreservierung.

Um ein Node-File zu erstellen, muss im CREATE-FILE explizit FILE-TYPE=*NODE-FILE angegeben werden. Im anderen Fall wird eine Net-Storage-Datei als BS2000-Datei angelegt.

Bestimmte Dateieigenschaften, wie z.B. die Zugriffsmethode bei Dateierstellung, Block- und Satzlänge, werden beim ersten Öffnen der Datei eingetragen. Werte zur logischen Dateigröße usw. werden erst beim Schließen der Datei übernommen. Mit dem Dienstprogramm SPCCNTRL (Space-Control) und dem Kommando SHOW-FILE-ATTRIBUTES können die Katalogeinträge ausgegeben werden.

Dateien und Dateigenerationen auf privaten Datenträgern und Dateien auf Net-Storage-Volumes, deren Katalogeintrag gelöscht wurde oder die in anderen Systemen erstellt wurden, können mit den Makros IMPORT oder FILE, Operand STATE=FOREIGN, bzw. mit dem Kommando IMPORT-FILE katalogisiert (= importiert) werden. Dabei muss für Plattendateien nur der erste Datenträger bereitgestellt werden, da der Katalogeintrag aus dessen F1-Kennsatz bzw. dem Katalogeintrag eines Net-Storage-Volumes erzeugt wird. Node-Files werden mit dem Makro IMPNFIL bzw. mit dem Kommando IMPORT-NODE-FILE katalogisiert. Für Banddateien muss die Folge der Datenträger im Makro- oder Kommandoaufruf beschrieben werden (siehe Abschnitt „Nicht-katalogisierte Banddateien“).

Eine Datei kann nur dann importiert werden, wenn für ihre Benutzerkennung ein Eintrag im Benutzerkatalog existiert. Ist die Benutzerkennung der Datei eine andere als die des laufenden Auftrags, muss sie mehrbenutzbar sein (Ausnahme bei Miteigentümerschaft, siehe "Festlegung einer Miteigentümerschaft (Co-Owner)"). Wird eine Datei aus einem anderen System oder von einer anderen Benutzerkennung übernommen, sollten die alten Katalogeinträge für diese Datei gelöscht werden.

5.1.2 Katalogeintrag aktualisieren

Bei jeder Dateiverarbeitung wird der Katalogeintrag in den Feldern aktualisiert, die sich aufgrund der Verarbeitung geändert haben. Das sind z.B. bei Eingabedateien die Felder ACC-DATE (Datum des letzten Zugriffs) und ACC-COUNT (Zugriffszähler). Für Ausgabedateien werden die Datei- und Verarbeitungseigenschaften in den Katalogeintrag übernommen sowie die Datenträgerliste, die Liste der Extents und die Dateigröße (Plattendateien: HIGH-US-PA; Banddateien: BLK-COUNT = Blockzähler).

Für Plattendateien kann die Speicherplatzzuweisung jederzeit mit dem Makro FILE (Operand SPACE) oder dem Kommando MODIFY-FILE-ATTRIBUTES verändert werden, sofern die Datei nicht gesperrt ist. Die belegten Seiten können nicht freigegeben werden. Sofern die Datei nicht gesperrt ist, lassen sich für Plattendateien die Schutzmerkmale mit dem Makro CATAL oder dem Kommando MODIFY-FILE-ATTRIBUTES ändern. Bei Banddateien werden die Schutzmerkmale beim ersten Öffnen der Datei in die Kennsätze übernommen und können daher später nicht beliebig verändert werden.

5.1.3 Informationen aus dem Katalogeintrag

Mit dem Makro FSTAT oder dem Kommando SHOW-FILE-ATTRIBUTES kann ein Benutzer jederzeit Informationen aus den Katalogeinträgen seiner Dateien – permanenter wie temporärer – abrufen. Ebenso kann er sich über die Katalogeinträge mehrbenutzbarer Dateien anderer Benutzer informieren.

Mit den verschiedenen Funktionen des Makro- und Kommandoaufrufs wird die Informationsausgabe gesteuert:

- Ausschnitte aus dem Katalogeintrag wählen
- Auswahlkriterien nutzen, die die Informationsausgabe in Abhängigkeit von im Katalogeintrag hinterlegten Merkmalen steuern
- die Ausgabe auf verschiedene Medien lenken: SYSOUT, SYSLST, Datei, Drucker

Auch das Dienstprogramm SPCCNTRL informiert über Katalogeinträge (siehe Handbuch „Dienstprogramme“ [13]).

5.1.4 Katalogeintrag löschen

Der Katalogeintrag wird automatisch gelöscht, wenn eine Datei gelöscht wird. Das Löschen von Dateiinhalt und Katalogeintrag kann jedoch durch die verschiedenen Funktionen des entsprechenden Makro- oder Kommandoaufrufs entkoppelt werden. Löschen des Dateiinhalts ohne Löschen des Katalogeintrags und ohne Speicherplatzfreigabe wird auch als „logisches Löschen“ bezeichnet (siehe "Katalogeintrag löschen"), Löschen des Katalogeintrags ohne Speicherplatzfreigabe als „Datei-Export“ (siehe dazu "Katalogeintrag löschen").

Die folgenden Tabellen zeigen Funktionsübersichten des Makros ERASE und der Kommandos DELETE-FILE und EXPORT-FILE zum Löschen von Katalogeinträgen.

Makro ERASE

Operanden	Version des Makro-Aufrufs	Daten-zerstörung 1)	Katalogeintrag löschen	belegten Speicherplatz freigeben	logisch löschen 2)	Bemerkungen
„pfadname“	0 – 3		ja	ja		
SPACE-CATALOG	1 – 3		ja	ja		Standard
DESTROY	0 – 3	ja	ja	ja		physikalische Datenzerstörung
DATA	0 – 3				ja	die Datei wird „logisch gelöscht“
DATA-KEEP-ATTR	3				ja	datenbezogene Eigenschaften bleiben erhalten
SPACE	0 – 3			ja		nur für Pubset-Dateien
CATALOG	0 – 3		ja			„private“ Datei wird exportiert
CATALOG, VOLUME	0		(ja)	(ja)		Datenträger wird exportiert
DELETE-OR-EXPORT	1 – 3		(ja)	(ja)		Wirkung hängt vom Datenträgertyp ab

DELETE-OR-EXPORT, VOLUME	1 – 3		(ja)	(ja)		Datenträger wird exportiert

[1] wenn der Katalogeintrag DESTROY=NO enthält

[2] „logisch löschen“ = Last Page Pointer zurücksetzen

Tabelle 6: Funktionen des Makros ERASE

Kommando DELETE-FILE

Operanden	Daten-zerstörung [1]	Katalogeintrag löschen	belegten Speicherplatz freigeben	logisch löschen [2]	Bemerkungen
FILE-NAME		ja	ja		
OPTION=ALL		ja	ja		Standardfunktion
OPTION=DESTROY-ALL	ja	ja	ja		
OPTION=DATA				ja	die Datei wird „logisch gelöscht“
OPTION=SPACE			ja		nur für Pubset-Dateien
OPTION=DATA-KEEP-ATTRIBUTES				ja	datenbezogene Eigenschaften bleiben erhalten

[1] wenn der Katalogeintrag DESTROY=NO enthält

[2] „logisch löschen“ = Last Page Pointer zurücksetzen

Tabelle 7: Funktionen des Kommandos DELETE-FILE

Kommando EXPORT-FILE

Operand	Katalogeintrag löschen	Bemerkung
FILE-NAME	ja	Datei auf Privatplatte oder Net-Storage-Volume wird exportiert

Tabelle 8: Funktionen des Kommandos EXPORT-FILE

Kommando EXPORT-NODE-FILE

Operand	Katalogeintrag löschen	Bemerkung
FILE-NAME	ja	Node-File (auf Net-Storage-Volume) wird exportiert

Tabelle 9: Funktionen des Kommandos EXPORT-NODE-FILE

Node-Files werden mit dem Makro IMPNFIL bzw. mit dem Kommando IMPORT-NODE-FILE katalogisiert.

Datei löschen

Die Standardfunktion des Makros ERASE bzw. des Kommandos DELETE-FILE hängt vom Datenträgertyp ab. Für Plattendateien ist die Standardfunktion „Speicherplatz freigeben und Katalogeintrag löschen“; zusätzlich kann der frei werdende Speicherplatz überschrieben werden siehe Abschnitt „Datenschutz durch „Datenzerstörung“ (DESTROY-Option)“). Für Banddateien ist die Standardfunktion „Katalogeintrag löschen“, das entspricht einem Datenträger-Export.

Das Überschreiben von Plattendaten beim Löschen kann im Makro- und Kommandoaufruf vereinbart, aber auch schon vorher im Katalogeintrag festgelegt werden:

- Operand DESTROY=YES im CATAL-Makro
- Operand PROTECTION...DESTROY-BY-DELETE in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES

Datei „logisch löschen“

Katalogeintrag und Speicherplatzzuweisung bleiben erhalten, es wird nur der Zeiger auf die letzte „beschriebene“ PAM-Seite (Highest-Used-Page) auf 0 zurückgesetzt. Die Daten sind zwar noch vorhanden, der Benutzer kann sie jedoch nicht mehr lesen, da ihm der physikalische Zugriff auf den Datenträger nicht gestattet ist. Der Katalogeintrag einer logisch gelöschten Datei entspricht in den Feldern CRE-DATE und HIGH-US-PA dem einer mit FILE (Makro) bzw. CREATE-FILE (Kommando) neu erzeugten Datei.
Von „logisch gelöschten Dateien“ kann man auch dann sprechen, wenn eine Datei mit OPEN OUTPUT (wieder-) eröffnet und sofort wieder geschlossen wird: Der Benutzer kann auf den alten Dateiinhalt nicht mehr zugreifen. Sowohl im Fall OPEN OUTPUT als auch beim Löschen mit dem Operandenwert DATA-KEEP-ATTR bleiben aber die datenbezogenen Eigenschaften im Katalogeintrag erhalten.

Datei-Export

Der Katalogeintrag wird gelöscht, der Speicherplatz und die Daten bleiben erhalten. Auf diese Weise können Dateien auf privaten Datenträgern oder Net-Storage-Volumes aus einem System entfernt werden. Sie können zu einem beliebigen Zeitpunkt in dasselbe oder ein anderes System wieder importiert werden (Makro IMPORT bzw. mit dem Kommando IMPORT-FILE; siehe auch Abschnitt „Katalogeintrag erstellen“). Node-Files werden mit dem Makro IMPNF bzw. mit dem Kommando IMPORT-NODE-FILES importiert.

Dateien werden mit dem Makro ERASE (Operanden DELETE-OR-EXPORT oder CATALOG) oder dem Kommando EXPORT-FILE exportiert. Wird in Verbindung mit den Makro-Operanden oder dem Kommando der Operand VOLUME angegeben (nur für Plattendateien zulässig!), werden alle Dateien exportiert, die auf diesem Datenträger liegen.

5.2 Ablageort

- Anfordern von Speicherplatz
  - Besonderheiten bei Pubsets
  - Besonderheiten bei Privatplatten
- Ablageortbestimmung bei SM-Pubsets
  - Direktattributierung
  - Storage-Klassen
  - Physikalische Allokierung
  - Bestimmung des Präformats
  - Bestimmung des Dateiformats
  - Defaultierung der ablageort-relevanten Dateiattribute
- Ablageortbestimmung bei SF-Pubsets

5.2.1 Anfordern von Speicherplatz

Das Anfordern von Speicherplatz auf gemeinschaftlichen Platten kann über den SPACE-Operanden des Makros FILE bzw. der Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES beeinflusst werden.

> **i** Bei Dateien auf Net-Storage wird der SPACE-Operand akzeptiert und die FILE-SIZE im Kataologeintrag entsprechend versorgt. Der Speicherplatz für die Datei wird im File-System erst belegt, wenn die Datei mit Nutzdaten der entsprechenden Größe gespeichert wird.

Wird eine Datei mit dem Makro FILE oder dem Kommando CREATE-FILE ohne Angaben im SPACE-Operanden katalogisiert, treten die Standardwerte des Systems in Kraft. Wird eine Datei mit dem Makro CATAL katalogisiert, wird nur der Katalogeintrag erstellt; es erfolgt keine Speicherplatzzuweisung.

Mit dem SPACE-Operanden wird die Primär- und/oder Sekundärzuweisung festgelegt.

Eine Speicherplatzanforderung ist bis etwa 4TB möglich. Das entspricht der maximalen Kapazität eines Pubsets abzüglich verbrauchter Seiten für Verwaltungsdaten des Systems (TSOSCAT, Benutzerkatalog, F5-Label,...). Die theoretische maximale im Katalog darstellbare Dateigröße beträgt 2.147.483.647 PAM-Seiten.

Primärzuweisung

Für die Datei wird sofort Speicherplatz reserviert oder freigegeben. Der im SPACE-Operanden genannte Wert wird auf ein Vielfaches von „k“ gerundet.
(„k“ ist die Anzahl von 2 KB-Blöcken pro Unit, der kleinsten Verwaltungseinheit der Speicherplatzverwaltung; siehe "Anfordern von Speicherplatz").

- Ist dieser Wert im Makro positiv bzw. in den Kommandos mit den Operanden SPACE=*RELATIVE oder SPACE=*ABSOLUTE angegeben, reserviert das DVS die entsprechende Anzahl PAM-Seiten für die Datei.
- Ist der Wert im Makro negativ bzw. in den Kommandos mit dem Operanden SPACE=*RELEASE angegeben, gibt das DVS – soweit möglich – die entsprechende Anzahl nicht belegter PAM-Seiten frei.

Sekundärzuweisung

Im Gegensatz zur Primärzuweisung wird die Sekundärzuweisung nicht sofort bei Einrichten der Datei wirksam, sondern erst, wenn bei Dateierstellung oder -erweiterung der reservierte Speicherplatz nicht ausreicht. Das System erhöht dann die Speicherplatzzuweisung für die Datei automatisch um die für die Sekundärzuweisung vereinbarte Anzahl PAM-Seiten (Kommando SHOW-FILE-ATTRIBUTES, Ausgabefeld S-ALLOC).

Der Wert für die Sekundärzuweisung wird nach jeder erfolgten Erweiterung verdoppelt. Die Verdoppelung endet, wenn der im System eingestellte Maximalwert erreicht ist. Die vereinbarte Anzahl von PAM-Seiten wird, falls erforderlich, bei der Speicherplatzzuweisung auf ein Vielfaches des Rundungsfaktors „k“ aufgerundet. Dieser Vorgang kann sich wiederholen, bis der dem Benutzer insgesamt zur Verfügung stehende Speicherplatz ausgeschöpft ist.

Aufteilung in zusammenhängende Speicherbereiche (Extents)

Sowohl bei Primär- als auch bei Sekundärzuweisung von Speicherplatz versucht das DVS, zusammenhängende Speicherbereiche zu reservieren und eine Aufteilung der Datei auf verschiedene Platten zu vermeiden.
Die entstehenden Dateiabschnitte werden als „Extents“ bezeichnet. Mit dem Makro FSTAT oder dem Kommando SHOW-FILE-ATTRIBUTES wird darüber informiert, aus wie vielen Extents eine Datei besteht und auf welchen Datenträgern/Platten diese liegen.

> ¡ Dateien auf Net-Storage haben stets (nur) einen Extent.

Speicherplatz wird in Größen einer Allokierungseinheit (Unit) vergeben, der kleinsten Verwaltungseinheit der Speicherplatzverwaltung. Eine Unit umfasst eine bestimmte Anzahl von 2KB-Blöcken, wobei derzeit die Stückelungen 6K, 8K und 64K möglich sind.
Die Stückelung kann vom Systembetreuer bei der Pubset-Generierung festgelegt werden, wobei Abhängigkeiten zum Plattenformat bestehen. Ein Pubset aus Key-Platten kann nur die Allokierungseinheit 6K annehmen.

Die folgende Tabelle stellt den Zusammenhang zwischen dem Rundungsfaktor k, der Unitgröße und der korrespondierenden Anzahl der PAM-Seiten dar:

„k“	Unit-Größe in KB	Anzahl an PAM-Seiten
6	6	3
8	8	4
64	64	32

Das Zeitverhalten bei der Dateiverarbeitung kann beinflusst werden, indem eine starke Zergliederung der Dateien vermieden wird. Wird die Extentzahl zu hoch, können die Dateien z.B. durch Kopieren (Makro COPFILE oder Kommando COPY-FILE) „reorganisiert“ werden.

Physikalische Allokierung

Unter physikalischer Allokierung versteht man die gezielte Vorgabe eines Ablageortes in einer der folgenden Varianten:

1. Angabe des Volume-Sets
2. Angabe des Volumes
3. Absolutzuweisung
   Über die Absolutzuweisung kann bestimmt werden, welche und wie viele PAM-Seiten für eine Datei reserviert werden sollen, indem die Anfangsadresse und die Größe des zu reservierenden Bereichs im SPACE-Operanden angegeben wird.

> ¡ Absolutzuweisung ist für Net-Storage-Volumes nicht möglich.

Eine physikalische Allokierung ist für Privatplatten generell möglich (Möglichkeit b) und c)). Für Pubsets ist eine physikalische Allokierung in folgenden Fällen möglich:

- für Dateien auf gemeinschaftlichen Datenträgern, wenn der Pubset mit einem entsprechenden Attribut zur Verfügung gestellt wurde (siehe Ausgabe des Kommandos SHOW-MASTER-CATALOG-ENTRY ..., INFORMATION=*USER)
- für Dateien auf gemeinschaftlichen Datenträgern, wenn der Benutzer das Recht auf physikalische Allokierung hat (siehe Ausgabe des Kommandos SHOW-USER-ATTRIBUTES ..., INFORMATION=*PUBSET-ATTRIBUTES).
- für Arbeitsdateien in SM-Pubsets

Speicherplatzmangel

Kann das System eine Speicherplatz-Anforderung nicht erfüllen, verfährt es folgendermaßen:

- Die für eine Primärzuweisung aufgerufenen Makros oder Kommandos werden entweder abgewiesen, oder es findet eine Teilzuweisung statt.
- Bei einer Sekundärzuweisung kann es zum Programmabbruch kommen.

Das DVS weist eine Speicherplatzanforderung in folgenden Fällen ab:

- Mit der Anforderung würde das dem Benutzer auf diesem Pubset zur Verfügung stehende Public-Space-Limit überschritten und der Benutzer ist nicht berechtigt, sein Public-Space-Limit zu überschreiten (PUBLIC-SPACE-EXCESS=*NO im Benutzereintrag).
- Wegen Speicherplatzmangel auf dem betroffenen Pubset oder der Privatplatte ist eine weitere Speicherplatzzuweisung nicht möglich.
- Die Anforderung ist für Net-Storage erfolgt und die Benutzung von Net-Storage ist im Benutzereintrag nicht erlaubt (NET-STORAGE-USAGE=*NOT-ALLOWED). Standardmäßig ist die Benutzung von Net-Storage erlaubt, es kann aber kein Limit vergeben werden.

Wird durch die neue Anforderung die Zahl der Extents zu groß für die Extentliste im Katalogeintrag oder überschreitet die Speicherplatzanforderung die freie Kapazität der Privatplatte, nimmt das DVS eine Teilzuweisung vor.
Führt eine Speicherplatzanforderung zum Überschreiten der maximalen, im Katalogeintrag vereinbarten Dateigröße, so wird nur die maximal mögliche Teilreservierung vorgenommen.

5.2.1.1 Besonderheiten bei Pubsets

Bei Pubsets hat der Benutzer mit Ausnahme der physikalischen Allokierung keinen Einfluss darauf, auf welchen Platten der Speicherplatz zugewiesen wird (wenn der Pubset aus mehreren Platten besteht). Mit dem Makro FSTAT, dem Kommando SHOW-FILE-ATTRIBUTES oder dem Dienstprogramm SPCCNTRL kann man sich über die Verteilung der Datei auf einzelne Platten informieren.

Jeder Pubset enthält folgende Informationen:

- Eintrag für alle Benutzer, die auf diesen Pubset zugreifen dürfen; andere Benutzer haben keinen Zugang zu diesem Pubset.
- Für jeden Benutzer, der das Zugriffsrecht hat, ist eine obere Schranke, das Public-Space-Limit, festgelegt.
- Für jeden Benutzer, der das Zugriffsrecht hat, ist festgelegt, ob er sein Public-Space-Limit überschreiten darf (PUBLIC-SPACE-EXCESS=*NO im Benutzereintrag).

Ein Benutzer, der auf einen Pubset nicht zugreifen darf, kann auf diesem Pubset auch keinen Speicherplatz anfordern.

Hat ein Benutzer die Zugriffsberechtigung für einen Pubset, kann er solange Speicherplatz anfordern, bis sein Public-Space-Limit erschöpft ist. Nur wenn er sein Public-Space-Limit überschreiten darf, dann kann er auch darüber hinaus Speicherplatz anfordern.

> ¡ Speicherplatz, den ein Benutzer auf Net-Storage belegt, wird nicht auf sein Public-Space-Limit angerechnet.

Der Operator erhält eine Meldung, wenn das Public-Space-Limit überschritten wurde. Ist diese Grenze auch bei Auftragsende noch überschritten, gibt das DVS bei der Logoff-Behandlung eine entsprechende Meldung an den Benutzer aus.

Die Speicherplatzbeschränkung wird von der Systembetreuung festgelegt. Mit dem Kommando SHOW-USER-ATTRIBUTES können Sie sich darüber informieren.

Speicherplatzerweiterung

Reicht der angeforderte Speicherplatz nicht aus, kann er erweitert werden. Dies kann explizit z.B. über das Kommando /MODIFY-FILE-ATTRIBUTES ...,SUPPORT= *PUBLIC-DISK(SPACE=*RELATIVE(...)) oder den Makro FILE ...,SPACE=... bzw. implizit während der Dateiverarbeitung durch das DVS erfolgen.
Die Allokierungsstrategie des DVS ist daraufhin ausgerichtet, eine Extent-Zersplitterung möglichst zu vermeiden, d. h. Anschluss an den letzten belegten Extent zu bekommen.

Speicherplatzfreigabe

Mit dem Makro ERASE (Operand SPACE) bzw. mit dem Kommando DELETE-FILE (Operand OPTION=*SPACE) kann Speicherplatz freigegeben werden. Der Katalogeintrag der betroffenen Dateien bleibt erhalten, er hat die Form eines mit dem Makro CATAL oder dem Kommando CREATE-FILE erstellten Katalogeintrags.

Reservierter, aber nicht belegter Speicherplatz kann mit dem Makro FILE (Operand SPACE=<-n>) bzw. dem Kommando MODIFY-FILE-ATTRIBUTES (Operand SPACE=*RELEASE(...)) freigegeben werden. Gilt aus dem Katalogeintrag DESTROY=YES, wird so freigegebener Speicherplatz überschrieben. Da beim Überschreiben keine Unit-Grenzen berücksichtigt werden, kann es vorkommen, dass (rückwärts) bis zur letzten belegten Seite die Restdaten überschrieben werden.

5.2.1.2 Besonderheiten bei Privatplatten

Speicherplatz auf Privatplatten wird wie Speicherplatz auf gemeinschaftlichen Datenträgern mit den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES oder dem Makro FILE angefordert. Das DVS stellt die Privatplatte für die Dauer der Anforderungen der beteiligten Aufträge bereit.

Speicherplatz auf privaten Platten kann in beliebigem Umfang angefordert werden, nur begrenzt durch die Kapazität des Datenträgers.
Für Privatplatten können Sie die relative Speicherplatzzuweisung mit Primär- und Sekundärzuweisung nutzen. Sie können festlegen, auf welchem Datenträger der Speicherplatz reserviert werden soll. Mit der Absolutzuweisung haben Sie außerdem die Möglichkeit, zu bestimmen, welche PAM-Seiten reserviert werden sollen.

Speicherplatzerweiterung

Eine Speicherplatzerweiterung erreichen Sie über den Makro FILE oder das Kommando MODIFY-FILE-ATTRIBUTES. Gibt man im Operanden VOLUME keine Archivnummern an, versucht das System, den Speicherplatz auf dem letzten Datenträger der Datei zu reservieren. Reicht dieser nicht für die gesamte Speicherplatzanforderung aus, wird Speicherplatz auf anderen im Katalog eingetragenen privaten Platten reserviert.

Geben Sie im VOLUME-Operanden Archivnummern an, versucht das System, Speicherplatz auf den dort angegebenen Datenträgern zu reservieren. Reicht dieser Speicherplatz nicht aus, erfolgt eine Teilreservierung. Platten, deren Archivnummern zwar im Katalog eingetragen sind, die aber im aktuellen Makro- oder Kommandoaufruf nicht aufgeführt wurden, werden nicht berücksichtigt.

Soll eine Datei, die bisher auf nur einer Privatplatte katalogisiert ist, auf eine zweite Privatplatte erweitert werden, müssen die Operanden VOLUME, DEVICE und SPACE angeben werden.
Bei großen Zusatzanforderungen ist es günstiger, Speicherplatz in mehreren kleineren Einheiten anzufordern, die eher zu Zuweisungen auf verschiedenen Platten führen.
Erstreckt sich eine Datei über mehrere private Platten, so müssen zum OPEN-Zeitpunkt alle Datenträger montiert sein!

Speicherplatzfreigabe

Reservierter oder nicht belegter Speicherplatz kann mit dem Makro FILE (Operand SPACE=<-n>) bzw. mit dem Kommando MODIFY-FILE-ATTRIBUTES (Operand SPACE= *RELEASE) freigegeben werden. Gilt aus dem Katalogeintrag DESTROY=YES, wird so freigegebener Speicherplatz überschrieben, wobei die Privatplatte zum Zeitpunkt des Makro- oder Kommandoaufrufs verfügbar sein muss.

5.2.2 Ablageortbestimmung bei SM-Pubsets

Die Situation, dass in einem Rechenzentrum mehrere SF-Pubsets mit unterschiedlichen Eigenschaften zur Verfügung gestellt werden, auf welche die Benutzer ihre Dateien verteilen, korrespondiert in einem SMS-Umfeld mit einem SM-Pubset, der Volume-Sets mit unterschiedlichen Eigenschaften enthält. Wie bei SF-Pubsets bestimmt die vorgegebene Katalogkennung (Catid) im Pfadnamen der Datei den Pubset. Der Auswahl eines geeigneten SF-Pubsets entspricht in einem SM-Pubset die Bestimmung eines geeigneten Volume-Sets innerhalb des SM-Pubsets.

In einem SM-Pubset werden durch Direktattributierung, Storage-Klassen und physikalische Allokierung verschiedene Instrumente zur Verfügung gestellt, den Ablageort der Dateien zu beeinflussen. Sie können innerhalb eines SM-Pubsets nebeneinander benutzt werden:

- Bei der Direktattributierung werden die Benutzer-Anforderungen anhand der einzelnen Dateiattribute Performance, Struktur, Verfügbarkeit, Arbeitsdatei spezifiziert. Die Auswahl des für die Datei geeigneten Volume-Sets erfolgt durch das System.
- Storage-Klassen sind benannte Kombinationen von Einzelattributen. Sie bieten einen größeren Komfort bei der Spezifikation der gewünschten Dateiattribute.
  Wie bei der Direktattributierung erfolgt bei der Nutzung von Storage-Klassen die Volume-Set-Auswahl durch das System.
- Die physikalische Allokierung erlaubt eine gezielte Vorgabe des Ablageorts innerhalb der gemeinschaftlichen Platten eines SM-Pubsets.

Es wird zunächst ein Umfeld beschrieben, in dem ausschließlich die direkte Dateiattributierung verwendet wird. Anschließend werden die Storage-Klassen als Erweiterung der direkten Dateiattributierung vorgestellt. Zum Schluss wird gezeigt, wie auch die physikalische Allokierung in die Nutzung eines SM-Pubsets integriert werden kann.

5.2.2.1 Direktattributierung

Zur Spezifikation der Benutzerwünsche für Performance, Verfügbarkeit und Dateistruktur werden die bereits für SF-Pubsets vorhandenen Dateiattribute benutzt.

Damit die Möglichkeiten eines SM-Pubsets effizient genutzt werden können, wird durch das System das Benutzerkommando SHOW-PUBSET-FILE-SERVICES zur Verfügung gestellt, welches die in einem SM-Pubset verfügbaren Services auf logischer Ebene anzeigt. Es informiert darüber, welche Kombinationen der ablageort-relevanten Dateiattribute AVAILABILITY, FORMAT, PERFORMANCE, USAGE, DISK-WRITE (und WORK-FILE) auf dem SM-Pubset optimal erfüllbar sind, welche Kombinationen nicht möglich sind, und welche Kombinationen erlaubt sind, aber nicht optimal realisiert werden können. Bei der Ermittlung der verfügbaren Services geht die aktuelle Belegungssituation der einzelnen Volume-Sets nicht ein, ebenso bleiben Benutzerberechtigungen und Benutzerkontingente unberücksichtigt.

Folgende, ablageort-relevante Dateiattribute spezifizieren die Benutzer-Anforderungen:

Präformat

Die Bedeutung der Strukturattribute (K, NK2, NK4) bleibt für SM-Pubsets im Vergleich zu SF-Pubsets unverändert. Es bestehen jedoch Unterschiede in den Mechanismen, über die einer Datei die Strukturattribute zugewiesen werden. Insbesondere ist es bei SM-Pubsets wichtig, dass bereits zum Zeitpunkt der Speicherplatzbereitstellung für eine neue bzw. eine schon katalogisierte Datei, der aber kein Speicherplatz zugeordnet war, Information bezüglich ihres Formats bereitgestellt werden kann. Damit kann diese bereits bei der Auswahl des Volume-Sets berücksichtigt werden.
Hierfür ist das Dateiattribut PREFORMAT vorgesehen, das einer Datei durch die Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES (Operand FILE-PREFORMAT) bei der erstmaligen Speicherplatzbereitstellung zugewiesen wird (Anmerkung: Als erstmalige Speicherplatzbereitstellung gilt auch der Fall, dass für eine bisher nur katalogisierte Datei Speicherplatz allokiert wird). Wird eine Datei über den FILE-Makro angelegt, so wird das Präformat unter anderem aus den dabei (d.h. zum Zeitpunkt der Speicherplatzbereitstellung) erfolgten Benutzerangaben für BLKCTRL und BLKSIZE abgeleitet.
Der Name „Präformat“ bringt zum Ausdruck, dass das darin spezifizierte Format noch nicht verbindlich ist. Die endgültige Formatfestlegung erfolgt für die Datei erst, wenn sie für ihre Erstellung geöffnet wird. Unter Umständen kann es sich zum OPEN-Zeitpunkt ergeben, dass der Volume-Set, auf dem der Datei bisher Speicherplatz zugewiesen war, mit dem gewünschten Dateiformat nicht verträglich ist. In diesem Fall erfolgt implizit durch die OPEN-Behandlung eine Verlagerung auf einen passenden Volume-Set, wenn ein solcher vorhanden ist. Um den System-Overhead gering zu halten und die Risiken bei einer möglichen Verlagerung (Speicherengpässe usw.) auszuschalten, wird dringend empfohlen, für Dateien eines SM-Pubsets zum Zeitpunkt der Speicherplatzbereitstellung das Präformat sinnvoll vorzugeben. Wenn der Benutzer für eine Datei auf eine explizite Vorgabe des Präformats verzichtet, wird es durch Defaultierung bestimmt. (Zum Defaultierungsverfahren siehe Abschnitt „Defaultierung der ablageort-relevanten Dateiattribute“.)

Performance

Die für SF-Pubsets eingeführten performance-relevanten Dateiattribute PERFORMANCE, USAGE, DISK-WRITE dienen auch zur Formulierung der Performance-Anforderung für Dateien auf SM-Pubsets. Allerdings ist die Bedeutung der performance-relevanten Dateiattribute bei SM- Pubsets allgemeiner, da sich die Attribute nicht mehr ausschließlich auf die Cache-Nutzung beziehen. Die statisch durch die Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES zugeordneten Attribute bewirken zunächst, dass für die Datei beim Anlegen ein geeigneter Volume-Set ausgesucht wird. Dies muss nicht zwingend ein mit einem Cache ausgestatteter Volume-Set sein, da erhöhte Performance auch mit anderen Möglichkeiten erreicht werden kann.
Die dynamische Modifizierung der performance-relevanten Dateiattribute PERFORMANCE und USAGE erlaubt die Reduzierung der statisch zugeordneten Werte für eine aktuelle Dateibearbeitung. Sie hat auf den Ablageort einer Datei keinen Einfluss und bezieht sich ausschließlich auf die Nutzung des Caches des Volume-Sets, auf dem die Datei zum OPEN-Zeitpunkt liegt. Ist dies ein Volume-Set, dem kein Cache zugeordnet ist, ist sie wirkungslos.

Verfügbarkeit

Zur Spezifizierung der Ausfallsicherheit wurde das Dateiattribut AVAILABILITY eingeführt (Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES; Operand AVAIL in den Makros FILE und CATAL). Im Gegensatz zu den Performance-Attributen wirkt es verbindlich, d.h. seine Erfüllung wird garantiert.
Für Arbeitsdateien und temporäre Dateien werden AVAILABILTY=*HIGH-Anforderungen zurückgewiesen. Ebenfalls abgewiesen wird der Auftrag, wenn kein geeigneter Volume- Set verfügbar ist oder die zulässigen Kontingente der Benutzerkennung überschritten werden. Belegt die Datei bereits Speicherplatz, der nur AVAILABILITY=*STD genügt, wird die Datei auf einen geeigneten Volume-Set verlagert.

Arbeitsdatei

Arbeitsdateien werden durch das Dateiattribut WORK-FILE bestimmt.

Voraussetzung dafür ist das Vorhandensein eines Work-Volume-Sets innerhalb des SM-Pubsets (siehe Kommando SHOW-PUBSET-DEFINITION-FILE). Zur Beschreibung von Arbeitsdateien siehe "Arbeitsdateien".

Voraussichtliche Dateigröße

Die voraussichtliche Größe beeinflusst die Volume-Set-Auswahl im Hinblick auf die Größe der Allokierungseinheit. Sie kann vom Benutzer nicht explizit vorgegeben werden, sondern wird implizit aus den Angaben der Werte für Primär- und Sekundärallokierung berechnet. Der Benutzer kann lediglich festlegen, ob seine Datei eine sog. „große Datei“ (Dateigröße >= 32 GB) werden darf oder nicht (siehe "Dateien größer 32 GB").

Gültigkeitsdauer der ablageort-relevanten Dateiattribute

Die ablageort-relevanten Dateiattribute Performance, Verfügbarkeit, Präformat und Arbeitsdatei werden einer Datei dann zugewiesen, wenn ihr erstmals Speicherplatz zugewiesen wird. Die Zuordnungen bleiben solange bestehen, bis sie durch den Benutzer geändert werden oder der Datei der Speicherplatz wieder vollständig entzogen wird. Für eine nur katalogisierte Datei, der kein Speicherplatz (auch nicht auf den Hintergrundebenen) zugewiesen ist, sind mit Ausnahme der Performance-Attribute die ablageort-relevanten Attribute undefiniert. Diese werden (erneut) aus Benutzerangaben ermittelt, wenn der Benutzer für die Datei (erneut) Speicherplatz anfordert (Kommando MODIFY-FILE-ATTRIBUTES).

Anpassung des Datei-Ablageorts an sich ändernde Anforderungen

Mithilfe des Kommandos MODIFY-FILE-ATTRIBUTES hat der Benutzer die Möglichkeit, für eine bestehende Datei die zugeordneten statischen Performance-Attribute PERFORMANCE, USAGE, DISK-WRITE sowie das Verfügbarkeitsattribut AVAILABILITY zu modifizieren und damit dem System Änderungen hinsichtlich der Performance- und Verfügbarkeitsanforderungen mitzuteilen. Die Änderung des Präformats ist nicht möglich. Bei der Erhöhung der gewünschten Verfügbarkeit ist zu beachten, ob das Servicespektrum des SM-Pubsets dies gestattet. Für die neu zugewiesenen Performance-Attribute bzw. Verfügbarkeitsattribute kann der Volume-Set, auf dem sich die Datei bisher befindet, sich als nicht mehr optimal geeignet oder als unverträglich erweisen.

Bei Unverträglichkeit der neuen Dateiattribute mit dem bisherigen Volume-Set wird implizit eine unmittelbare Verlagerung auf einen verträglichen Volume-Set durchgeführt, falls ein solcher vorhanden ist. Bei nicht optimaler Eignung bleibt die Datei zunächst auf dem bisherigen Volume-Set, auch wenn vorübergehend Benutzeranforderungen nicht voll befriedigt oder Resourcen nicht optimal genutzt werden. Dass sich letztlich wieder eine günstige Verteilung der Dateien auf die einzelnen Volume-Sets einstellt, muss der Systembetreuer durch Maßnahmen der Pubset-Reorganisation ermöglichen. Eine implizite Reorganisation ergibt sich immer beim Zurückholen einer Datei von einer Hintergrundebene in die Verarbeitungsebene, da bei dieser Gelegenheit ein Volume-Set auf Basis der aktuellen Dateiattribute neu ermittelt wird.

Die Aktivitäten zum Verlagern der Dateien bleiben dem Benutzer als Maßnahmen auf physikalischer Ebene (weitgehend) verborgen. Insbesondere haben sie keinen Einfluss auf die Adressierung der Dateien und erfordern keine Anpassungen der Benutzer-Prozeduren.

5.2.2.2 Storage-Klassen

Eine Storage-Klasse repräsentiert eine bestimmte Wertekombination der ablageort-relevanten Dateiattribute AVAILABILITY, PERFORMANCE, USAGE, DISK-WRITE, FILE-PREFORMAT und WORK-FILE. Sie kann außerdem durch den Systembetreuer mit einer vordefinierten Liste von Volume-Sets verknüpft werden, die bei der Speicherplatzbereitstellung für eine Datei durch das System vorrangig (nicht zwingend!) berücksichtigt wird. Die Benutzung von Storage-Klassen kann durch die Zuordnung von Guards-Profilen eingeschränkt werden.

Über die vom Systembetreuer eingerichteten und für die Benutzer erlaubten Storage-Klassen und deren Eigenschaften (mit Ausnahme der Volume-Set-Listenzuordnung) informiert das Kommando SHOW-STORAGE-CLASS. Durch den Operand STORAGE-CLASS der Kommandos CREATE-FILE und MODIFY- FILE-ATTRIBUTES bzw. den Operand STOCLAS in den Makros FILE und CATAL kann einer Datei eine geeignete Storage-Klasse (an Stelle von Direktattributen) zugeordnet und damit gegenüber dem System die Anforderungen für diese Datei spezifiziert werden.

Storage-Klassen bieten gegenüber Direktattributierung die Möglichkeit der Bereitstellung einer komfortablen, für das Serviceangebot eines bestimmten SM-Pubsets maßgeschneiderten Benutzerschnittstelle. Durch die Benutzung von Storage-Klassen an Stelle von Direktattributierung können JCL-Prozeduren kompakt und übersichtlich gehalten werden.

Storage-Klassen sind pubset-spezifisch definiert.

Der Benutzer muss sich für die Nutzung der in einem SM-Pubset vorhandenen Möglichkeiten über die verfügbaren Storage-Klassen und ihre Bedeutung informieren, Kenntnisse über die konkrete physikalische Konfiguration, welche die Services realisieren, benötigt er nicht.

Änderung der Service-Anforderung für eine Datei

Einem sich ändernden Anforderungsprofil einer Datei kann der Benutzer durch das Zuordnen einer neuen Storage-Klasse Rechnung tragen. Die Zuordnung einer neuen Storage-Klasse bewirkt, dass – mit Ausnahme des Präformats – die in der Definition der Storage-Klasse festgelegten Dateiattribute auf die Datei übertragen werden. Durch die Änderung des Anforderungsprofils einer Datei kann es sich ergeben, dass der Volume-Set, auf dem die Datei bisher liegt, nicht mehr günstig ist, weil er die geänderten Dateiattribute nicht oder nur unzureichend erfüllt oder er nicht in der Volume-Set-Liste der neu zugeordneten Stora-ge-Klasse enthalten ist.

Eine unmittelbare Verlagerung der Datei auf einen anderen Volume-Set erfolgt, wenn die neuen Dateiattribute mit dem bisherigen Ablageort der Datei nicht verträglich sind.
Eine unmittelbare Verlagerung kann auch erfolgen, wenn der Volume-Set, zu dem die Datei gehört, nicht zu der Volume-Set-Liste der neuen Storage-Klasse gehört. Falls keine unmittelbare Verlagerung erfolgt, werden die geänderten Anforderungen an die Datei erst beimnächsten Recall von einer Hintergrundebene in die Verarbeitungsebene wirksam. Dies kann z.B. im Rahmen einer vom Systembetreuer initiierten Maßnahme für die Pubset-Reorganisation geschehen. Der Benutzer kann für eine Datei, der eine Storage-Klasse zugeordnet ist, auch die einzelnen ablageort-relevanten Attribute durch Verwendung von Direktattributierung ändern. Dabei wird die bisher bestehende Storage-Klassen-Zuordnung aufgehoben. Dies muss der Benutzer vor allem dann beachten, wenn es sich um eine Storage-Klasse handelt, der eine Volume-Set-Liste zugeordnet ist.

Änderung der Eigenschaften einer Storage-Klasse

Die Änderungen, die sich auf Volume-Set-Listen beziehen, kommen automatisch auch für bestehende Dateien zum Tragen, da sie bei späteren Recalls von Hintergrundebenen berücksichtigt werden. Die Änderung der Attributkombination einer Storage-Klasse überträgt sich hingegen nicht automatisch auf bestehende Dateien, ein Update kann aber durch den Anwender (Benutzer oder Systembetreuer) explizit durch das Kommando MODIFY-FILE-ATTRIBUTES (STORAGE-CLASS=*UPDATE) veranlasst werden.

5.2.2.3 Physikalische Allokierung

Das Privileg TSOS berechtigt generell zur physikalischen Allokierung auf allen Benutzerkennungen. Der Systembetreuer kann einzelnen Benutzerkennungen das Recht einräumen, dass Dateien dieser Kennung physikalisch allokiert werden dürfen (siehe auch „„Physikalische Allokierung“ (Anfordern von Speicherplatz)). Für die physikalische Allokierung sind in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES die Operanden VOLUME-SET, VOLUME und SPACE=*ABSOLUTE(..) vorgesehen.
Der Operand VOLUME-SET ist nur für SM-Pubsets relevant.

Folgende Detaillierungslevel sind möglich:

1. physikalische Allokierung auf Block-Ebene
   Der Aufrufer spezifiziert sowohl das Volume als auch die Blöcke innerhalb dieses Volumes, auf dem Speicherplatz bereitgestellt werden soll.
2. physikalische Allokierung auf Volume-Ebene
   Der Aufrufer spezifiziert, auf welchen Volumes des Pubsets die Datei abzulegen ist. Es ist für ihn ohne Interesse, wo innerhalb der Volumes der Speicherplatz bereitgestellt wird.
3. physikalische Allokierung auf Volume-Set-Ebene
   Der Aufrufer spezifiziert, auf welchem Volume-Set die Datei abzulegen ist. Es ist für ihn ohne Interesse, wo innerhalb des Volume-Sets der Speicherplatz bereitgestellt wird.

> ¡ Physikalische Allokierung ist für Net-Storage-Volumes auf Volume-Ebene möglich. Das Privileg zur physikalischen Allokierung ist in diesem Fall nicht erforderlich.

Informationsfunktionen für Benutzer der physikalischen Allokierung

Für die Benutzer der physikalischen Allokierung stehen Informationsfunktionen zur Verfügung, welche die Sicht auf die physikalische Konfiguration ermöglichen:

- Das Kommando SHOW-PUBSET-DEFINITION-FILE gibt detailliert Auskunft über die Pubset-Konfiguration und die Pubset-Eigenschaften von SM-Pubsets.

  Für die einzelnen Volumes zeigt es Nutzungsart (STANDARD, HSMS-CONTROLLED, WORK), Allokierungsbeschränkungen (NEW-FILE-ALLOCATION), sowie Performance-Eigenschaften an. Zur Darstellung der Performance- und die Verfügbarkeitscharakteristik werden die vom Systembetreuer vorgenommenen Definitionen herangezogen, eine weitergehende Beschreibung der zu Grunde liegenden Physik (Cache-Typ, DRV) erfolgt nicht.
- Das Kommando SHOW-PUBSET-RESTRICTION zeigt die Volume-Sets des SM-Pubsets, die ihnen zugehörenden Volumes, sowie – analog wie bei SF-Pubsets – den Gerätetyp und die Allokierungsbeschränkungen der einzelnen Volumes an.

Beispiel

```
/SHOW-PUBSET-DEFINITION-FILE PUBSET=SMS1,VOLUME-SET=VS01  ──────────────────────── (1)
------------------------------------------------------------------------------
COMMAND: SHOW-PUBSET-DEFINITION-FILE
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PUBSET SMS1: TYPE = SYSTEM-MANAGED, VOLUMESETS = 3, DEFAULT FILE FORMAT = NK4
---- VOLUME-SET INFORMATION --------- + -----------------------------------
VOLUME-SET VS01: NORMAL-USE, NK4-FORMAT
---- GLOBAL ATTRIBUTES      --------- + -----------------------------------
 AVAILABILITY                         | STANDARD
 USAGE                                | STANDARD
 FORMAT                               | NK4-FORMAT
 MAXIMAL I/O LENGTH                   | 48  HP
 ALLOCATION UNIT SIZE                 | 4   HP
 DRV-VOLSET                           | NO
 NEW FILE ALLOCATION                  | NOT RESTRICTED
 VOLUME SET ACCESS                    | NOT RESTRICTED
---- PERFORMANCE ATTRIBUTES --------- + -----------------------------------
 PERFORMANCE                          | STANDARD
 WRITE-CONSISTENCY                    | BY-CLOSE
------------------------------------------------------------------------------
```

(1) Angezeigt werden die physikalische Volume-Set- und Volume-Konfiguration des SM-Pubsets SMS1.

```
/SHOW-PUBSET-RESTRICTION PUBSET=SMS1 ──────────────────────────────────────────  (2)
------------------------------------------------------------------------------
COMMAND: SHOW-PUBSET-RESTRICTION
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
PUBSET SMS1: TYPE = SYSTEM-MANAGED, VOLUMESETS = 3, DEFAULT FILE FORMAT = NK4
---- PHYSICAL CONFIGURATION --------- + -----------------------------------
---- VOLUME-SET INFORMATION --------- + -----------------------------------
VOLUME-SET VS00: VOLUMES = 2
 VOLUME CONFIGURATION:
  VOLUME    DEVICE     ALLOCATION       VOLUME    DEVICE     ALLOCATION
  VS00.0    D3435      NOT RESTR        VS00.1    D3435      NOT RESTR
VOLUME-SET VS01: VOLUMES = 1
 VOLUME CONFIGURATION:
  VOLUME    DEVICE     ALLOCATION       VOLUME    DEVICE     ALLOCATION
  VS01.0    D3435      NOT RESTR
VOLUME-SET VS02: VOLUMES = 1
 VOLUME CONFIGURATION:
  VOLUME    DEVICE     ALLOCATION       VOLUME    DEVICE     ALLOCATION
  VS02.0    D3435      NOT RESTR
------------------------------------------------------------------------------
```

(2) Angezeigt werden die globalen Volume-Set-Eigenschaften (Verfügbarkeit, Nutzungsart, Allokierungs-Restriktionen usw.) und die Performance-Attribute(Performance-Profil und Schreibsicherheit) des Volume-Sets VS01 im SM-Pubset SMS1.

Das Kommando SHOW-MASTER-CATALOG-ENTRY liefert bei SM-Pubset-Einträgen Informationen über pubset-globale Zustände (z.B. accessible, inaccessible, local, remote, siehe Handbuch „HIPLEX MSCF“ [11]). Bei Einträgen für Volume-Sets wird der zugehörige Pubset angezeigt. Bei der STAMCE-Makroschnittstelle muss ebenfalls die Struktur von SM-Pubsets beachtet werden:

- Pubset-globale Zustände (accessible, inaccessible, usw.) werden wie bei SF-Pubsets über den MRSCAT-Eintrag des Pubsets angezeigt.
- Konfigurationsdaten bezüglich Cache-Zuordnungen, Struktur, Format usw. sind bei SM-Pubsets volume-set-spezifisch und finden sich daher in den MRSCAT-Einträgen der Volume-Sets. Dort ist auch hinterlegt, welchem Pubset der Volume-Set angehört.

Auskunftsfunktionen bis zum Detaillierungsgrad von einzelnen Blöcke werden auch für SM-Pubsets durch das Dienstprogramm SPCCNTRL (Funktion DISPLAY) angeboten, siehe Handbuch „Dienstprogramme“ [13].

Berücksichtigung von Migrationssperren

Die Art der Ablageort-Fixierung einer Datei wird durch die Dateiattribute S0-MIGRATION und MIGRATE beschrieben. Diese Attribute können einer Datei durch den Benutzer explizit z.B. mit den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES zugewiesen werden. Sie dienen als Schnittstelle zwischen dem Benutzer und dem Systembetreuer bzw. dem System.

Sie erlauben, dass bei den i.A. durch den Systembetreuer organisierten Aktivitäten „Pubset-Reorganisation“ und „Dateiverdrängung auf Hintergrundebenen“ (Migration; durch das Produkt HSMS) die speziellen Anforderungen benutzerkontrollierter Dateien erkannt und auf einfache Weise berücksichtigt werden können. Dies geschieht z.B dadurch, dass HSMS Dateien mit MIGRATE-FORBIDDEN von der Migration auf Hintergrundebenen ausschließt und auf Hintergrundebenen befindliche Dateien mit S0-MIGRATION-FORBIDDEN beim Recall auf ihren Herkunfts-Volume-Set zurückbringt.

> ¡ Dateien auf Net-Storage werden generell nicht auf eine Hintergrundebene migriert.

Physikalisch allokierte Dateien und Migrationssperren

Die physikalische Allokierung steht in engem Zusammenhang mit der Bestimmung des Ablageorts und muss bei Dateiverlagerungen (über das Produkt HSMS) innerhalb der S0-Ebene sowie zwischen S0-Ebene und Hintergrundebenen beachtet werden. Die physikalische Allokierung ist im Allgemeinen direkt mit dem Wunsch einer bestimmten Ablageort-Fixierung gekoppelt. Erfolgt die erstmalige Speicherplatzanforderung (also nicht beim Erweitern einer bestehenden Datei!) über physikalische Allokierung und werden die Fixierungs-Attribute dabei nicht explizit vorgegeben, werden für S0-MIGRATION und MIGRATE implizite Voreinstellungen vorgenommen. Die voreingestellten Werte sind abhängig vom Detaillierungslevel, auf dem der Ablageort vorgegeben wird:

- Bei Angabe auf Volume-Set-Ebene (Operand VOLUME-SET) wird S0-MIGRATION= *FORBIDDEN sowie MIGRATE=*ALLOWED eingestellt. Es wird davon ausgegangen, dass die Datei während der Verarbeitungsphase in der S0-Ebene immer auf dem vorgegebenen Volume-Set liegen soll. Die Lage innerhalb des Volume-Sets kann jedoch verändert werden. Daher darf die Datei im Rahmen von Dateiverdrängung oder einer Pubset-Reorganisation auf Hintergrundebenen migriert werden; es wird aber sichergestellt, dass sie bei einem späteren Recall wieder auf den alten Volume-Set gelangt.
- Bei physikalischer Allokierung auf Volume- oder Blockebene erfolgt die Voreinstellung S0-MIGRATION=*FORBIDDEN und MIGRATE=*FORBIDDEN. In diesem Fall ist keine Migration auf eine Hintergrundebene möglich, die Lage der Datei bleibt so, wie sie beim Anlegen spezifiziert wurde.

Wie bereits erwähnt wurde, können Migrationsperren auch explizit modifiziert werden.
Da dadurch ähnlich starker Einfluss auf die Speicherplatznutzung eines Pubsets genommen werden kann wie durch physikalische Allokierung, sind die Einstellungen S0-MIGRATION=*FORBIDDEN und MIGRATE=*FORBIDDEN an die Berechtigung zur physikalischen Allokierung geknüpft.

Zusammenhang von physikalischer Allokierung und Dateiattributen

Durch Angabe der Attribute zu Performance, Verfügbarkeit, Format und Arbeitsdatei werden die Anforderungen an eine Datei spezifiziert. Das System trägt diesen Anforderungen bei der Auswahl eines geeigneten Volume-Sets Rechnung. Bei physikalischer Allokierung sind für die Ablageortbestimmung die Dateiattribute ohne Relevanz, da in diesem Fall der Benutzer selbst den Volume-Set bestimmt.
Die Dateiattribute erfüllen aber noch weitere Funktionen, die auch bei physikalisch allokierten Dateien nicht entfallen:

- Ist ein Volume-Set mit einem Cache ausgestattet, so steuern die performance-bezogenen Dateiattribute die Cache-Nutzung;
- Der Volume-Set, auf dem eine Datei liegt, legt im Allgemeinen nicht eindeutig das Format der auf ihm liegenden Dateien (K, NK2, NK4) fest. Zur genauen Format-Spezifikation sind zusätzliche Angaben erforderlich.
- Über die Dateiattribute wird ermittelt, welche Benutzerkontingente durch den von einer Datei belegten Speicherplatz belastet werden.

Als Konsequenz werden einer Datei auch beim Anlegen durch physikalische Allokierung für die Dateiattribute PERFORMANCE, IO-USAGE, DISK-WRITE, AVAILABILITY, PREFORMAT, USAGE gültige Werte zugewiesen. Allerdings bestehen zwischen physikalischer Allokierung und nicht physikalischer Allokierung Unterschiede sowohl hinsichtlich der Defaultierung der Werte als auch in der Art der Berücksichtigung von expliziten Benutzerangaben:

- Attribute, die bereits durch die Vorgabe des Volume-Sets fixiert sind, werden berücksichtigt und entsprechend eingetragen.
- Falls vom Benutzer spezifizierte Attribute mit den Eigenschaften des vorgegebenen Volume-Sets unverträglich sind, wird die Anforderung zurückgewiesen.
- Storage-Klassen sind in Kombination mit physikalischer Allokierung nicht nutzbar.

Zusammenhang zwischen S0-Migrationssperre und Dateiattributen

Ähnlich wie bei der physikalischen Allokierung ist auch beim Ändern der Attribute einer Datei, für welche die S0-Migrationssperre gesetzt ist, sowie bei dem Setzen der Migrationssperre für eine Datei zu verhindern, dass die Kontingentüberwachung unterlaufen werden kann. Daher werden auch in diesen Fällen gegebenfalls automatisch Modifikationen der bisherigen bzw. der vom Benutzer neu vorgegebenen Attribute vorgenommen. Legt z.B. ein Benutzer eine Datei mit dem Attribut AVAILABILITY=*HIGH an, und nimmt anschließend durch ein MODIFY-FILE-ATTRIBUTES-Kommando die Modifikation AVAILABILITY=*STD, S0-MIGRATION=*FORBIDDEN vor, so wird implizit die Datei wieder als hoch verfügbar attributiert.

5.2.2.4 Bestimmung des Präformats

Die Bestimmung des Dateiformats (K, NK2, NK4) ist bei SM-Pubsets komplexer als bei SF-Pubsets. Bei der Erstallokierung wird für eine optimale Ablageortbestimmung vom System ein Präformat für die Datei bestimmt und im Katalogeintrag abgelegt. Das tatsächliche Dateiformat wird erst beim Eröffnen der Datei aus dem Präformat, dem tatsächlichen Ablageort und den OPEN-Parametern festgelegt (siehe Abschnitt „Bestimmung des Dateiformats“).

Wie bereits im Abschnitt „Direktattributierung“ erwähnt, ist es jedoch ratsam, das gewünschte Dateiformat bereits bei der Erstallokierung von Speicherplatz dem System mitzuteilen. Welche Möglichkeiten es hierfür gibt und welche Defaultierungsmechanismen vom System bestehen, soll dieser Abschnitt zeigen.

Es gibt zwei globale Defaultierungsmechanismen, die vom System angeboten werden:

- das Pubset-Default-Format
- die benutzer-spezifische Default-Storage-Klasse

Pubset-Default-Format

In Analogie zum SF-Pubset, bei dem das Pubset-Format implizit durch das Format seiner Platten vorgegeben ist, wird für SM-Pubsets ein Default-Format (durch den Systembetreuer bei der Pubset-Generierung) festgelegt. Dieses Default-Format kann sich im Laufe einer Pubset-Session ändern, wenn dies durch Rekonfigurationsmaßnahmen seitens des Systembetreuers notwendig wird.

Über das Pubset-Default-Format informieren die Kommandos SHOW-PUBSET-DEFINITION-FILE und SHOW-PUBSET-FILE-SERVICES PUBSET=..., FILE-FORMAT=*BY-PUBSET-DEFAULT bzw. der Makro STAM.

Benutzer-spezifische Default-Storage-Klasse

Der Systembetreuer hat die Möglichkeit, für jeden Benutzer auf jedem SM-Pubset eine Default-Storage-Klasse festzulegen. Der in dieser Storage-Klasse eingetragene Wert des FILE-PREFORMAT (oder der daraus resultierende Wert bei *BY-PUBSET-DEFAULT) wird in bestimmten Fällen (siehe unten) in den Katalogeintrag der Datei übernommen.

Über das Vorhandensein einer benutzer-spezifischen Default-Storage-Klasse informiert das Kommando SHOW-USER-ATTRIBUTES PUBSET=...,INFORMATION=*PUBSET-ATTRIBUTES. Auskunft über den Inhalt der Storage-Klasse gibt das Kommando SHOW-STORAGE-CLASS.

Das Präformats lässt sich ableiten aus der vollständigen Direktattributierung oder unterschiedliche Defaultierungen:

1. Vollständige Direktattributierung

   Sie liegt vor, wenn der Benutzer eine Attributierung vornimmt, aus der sich direkt ein eindeutiges Präformat ableiten lässt. Dies ist in folgenden Fällen gegeben:

angegebene Attribute	**resultierendes FILE-PREFORMAT**
FILE-PREFORMAT=K	K
FILE-PREFORMAT=NK2	NK2
FILE-PREFORMAT=NK4	NK4
BLKCTRL=PAMKEY	K
BLKCTRL=DATA2K	NK2
BLKCTRL=DATA4K	NK4
BLKCTRL=DATA / NO und BLKSIZE=(STD,n) (n = ungerade)	NK2
BLKCTRL=DATA / NO und BLKSIZE=(STD,n) (n = gerade)	NK4

2. Defaultierungen

   Falls keine vollständige Direktattributierung vorliegt, wird die Bestimmung des Präformats über unterschiedliche Defaultierungen vorgenommen. Die Reihenfolge der verschiedenen Mechanismen ist in folgender Tabelle wiedergegeben:

Benutzerangaben	**resultierendes FILE-PREFORMAT**
1. physikalische Allokierung	gemäß Volume-Set-Eigenschaft [1)]
2. Angabe einer Storage-Klasse mit FILE-PREFORMAT ungleich *BY-PUBSET-DEFAULT	gemäß Präformat in der angegebenen Storage-Klasse [1)]
3. keine Storage-Klassen-relevanten Attribute [2)] angegeben und eine Default-Storage-Klasse mit FILE-PREFORMAT ungleich *BY-PUBSET-DEFAULT	gemäß Präformat in der Default-Storage-Klasse [1)]
4. sonst	gemäß Pubset-Default-Format [1)]

1) Falls sich das Präformat als K ergibt und als BLKCTRL-Wert *DATA oder *NO angegeben ist, wird NK2 statt K eingestellt.

2) Eine Storage-Klassen-relevante Attributangabe liegt in folgenden Fällen vor:

- Kommandoschnittstelle (CREATE-FILE/MODIFY-FILE-ATTRIBUTES): wenn für STORAGE-CLASS der Wert *NONE oder einer der Operanden AVAILABILITY, DEVICE-TYPE, DISK-WRITE, VOLUME, VOLUME-SET, WORK-FILE, PERFORMANCE oder USAGE angegeben ist
- Makroschnittstelle (FILE): wenn für einen der Operanden IOPERF oder IOUSAGE ein Wert ungleich dem NULL-Operanden oder für STOCLAS der Wert *NONE oder einer der Operanden AVAIL, DEVICE, DISKWR, VOLUME, VOLSET oder WORKFIL angegeben ist

5.2.2.5 Bestimmung des Dateiformats

Das letztendlich einzustellende Dateiformat spiegelt sich in den Attributen BLOCK-CONTROL-INFO (bzw. BLKCTRL) und BUFFER-LENGTH (bzw. BLKSIZE) wider. Sie werden erst während dem Öffnen der Datei (Makro OPEN, Open-Modus OUTPUT bzw. OUTIN) bestimmt. Entscheidend für die Bestimmung sind:

- Die Angaben zu FCBTYPE, BLKCTRL und BLKSIZE, wobei die maßgeblichen Input-Werte während der OPEN-Verarbeitung aus den Informationsquellen TFT, FCB und Katalogeintrag ermittelt werden (siehe Kapitel „OPEN-Verarbeitung“)
- Das Format des tatsächlichen Ablageorts (K, NK2, NK4)
- Das Präformat: Bei Dateien mit CREATION-DATE=*NONE ist dies der Wert des Attributs FILE-PREFORMAT. Bei Dateien, die bereits ein CREATION-DATE besitzen, wird das aktuelle Format der Datei (abgeleitet aus den existierenden Attributen FCBTYPE, BLKSIZE und BLKCTRL) verwendet

Die angegebenen Attributwerte zur Bestimmung des Formats werden i.A. übernommen bzw. wie folgt defaultiert:

Präformat der Datei	Defaultierung von BLKSIZE
K / NK2	(STD,1)
NK4	(STD,2)

Präformat der Datei	FCBTYPE	Defaultierung von BLKCTRL
K	PAM / SAM / ISAM	PAMKEY
NK2 / NK4	SAM	DATA
NK2 / NK4	PAM	NO
NK2	ISAM	DATA2K
NK4	ISAM	DATA4K [1)]

1) Bei FCBTYPE=ISAM und BLKCTRL=DATA4K wird eine Angabe BLKSIZE=(STD,n) mit n ungerade abgewiesen. Eine Defaultierung erfolgt immer mit (STD,2).

Bei Unverträglichkeiten mit dem Ablageort oder Angaben, die nicht mit der Zugriffsmethode vereinbar sind, werden während der OPEN-Verarbeitung wie folgt Modifikationen vorgenommen:

	altes File-Präformat	neues File-Präformat
BLKCTRL=PAMKEY	NK2 / NK4	K
BLKCTRL ungleich PAMKEY und BLKSIZE=(STD,n) mit n ungerade	NK4	NK2

FCBTYPE	alter BLKCTRL-Wert	neuer BLKCTRL-Wert
SAM	DATA2K / DATA4K / NO	DATA
PAM	DATA2K / DATA4K	NO
ISAM	DATA / NO	• DATA4K, falls das resultierende Präformat NK4 ist • DATA2K sonst

5.2.2.6 Defaultierung der ablageort-relevanten Dateiattribute

Die Abbildung der vom Benutzer getroffenen Angaben für die Dateiattribute auf die tatsächlich resultierenden Werte erfolgt durch Defaultierungs-Mechanismen, welche u.a. vom Benutzer nicht explizit getroffene Vorgaben ergänzen und ggf. auch explizit spezifizierte Werte modifizieren. Folgende Defaultierungs-Mechanismen existieren:

- Defaultierung der einzelnen ablageort-relevanten Attribute über Storage-Klassen und Default-Storage-Klassen (Attribut STORAGE-CLASS)
- Defaultierung des Dateiformats (Attribut FILE-PREFORMAT)
- Defaultierung der Verfügbarkeit (Attribut AVAILABILITY)
- Defaultierung der performance-bezogenen Attribute (Attribut PERFORMANCE)
- Defaultierung des Dateityps Arbeitsdatei (Attribut WORK-FILE)
- Defaultierung des Attributs S0-MIGRATION
- Defaultierung des Attributs MIGRATE

Die Mechanismen werden exemplarisch anhand der Kommandoschnittstelle (hauptsächlich CREATE-FILE und MODIFY- FILE-ATTRIBUTES) erklärt. Für die Programmschnittstellen (Makros CATAL und FILE) gilt analoges.

Defaultierung der einzelnen ablageort-relevanten Attribute über Storage-Klassen und Default-Storage-Klassen

Storage-Klassen müssen durch den Systembetreuer definiert werden. Zur Bestimmung der ablageort-relevanten Dateiattribute WORK-FILE, PERFORMANCE, USAGE, DISK-WRITE, FILE-PREFORMAT und AVAILABILITY werden die in der Storage-Klasse hinterlegten Werte herangezogen, wobei der Performance-Wert unter Berücksichtigung der Benutzerberechtigung DMS-TUNING-RESOURCES ggf. durch das System reduziert wird.

Default-Storage-Klassen (benutzer- und pubset-spezifisch)

Der Systembetreuer hat die Möglichkeit, für jeden Benutzer auf jedem SM-Pubset eine Default-Storage-Klasse festzulegen. Bei jeder Erstallokierung von Speicherplatz auf diesem Pubset unter dieser Benutzerkennung werden dann die Attributeinstellungen dieser Storage-Klasse übernommen, wenn keine der folgenden Angaben erfolgt:

- explizite Angabe einer Storage-Klasse
- STORAGE-CLASS=*NONE (*)
- WORK-FILE=*NO *)
- WORK-FILE=*YES
- physikalische Allokierung
- PERFORMANCE *)
- USAGE *)
- DISK-WRITE *)
- AVAILABILITY *)

*) Wenn kein Recht auf physikalische Allokierung vorliegt, macht die Default-Storage-Klasse die mit *) gekennzeichneten Angaben unwirksam (und nicht umgekehrt).

Über das Vorhandensein einer benutzer-spezifischen Default-Storage-Klasse informiert das Kommando SHOW-USER-ATTRIBUTES PUBSET=..., INFORMATION=*PUBSET-ATTRIBUTES.

Für jede Dateigenerationsgruppe in einem SM-Pubset kann durch den Benutzer eine Default-Storage-Klasse vereinbart werden (Operand STOR-CLASS-DEFAULT im Kommando CREATE-FILE-GROUP). Bei der Erstallokierung von Speicherplatz einer Dateigeneration werden die Attributeinstellungen dieser Storage-Klasse unter den gleichen Bedingungen wie bei den Dateien übernommen. Die Default-Storage-Klasse einer Dateigenerationsgruppe kann explizit angegeben werden oder es wird die benutzer-spezifische Default-Storage-Klasse des SM-Pubsets zugewiesen (STOR-CLASS-DEFAULT=*STD), unter dem die Dateigenerationsgruppe angelegt wird.

Zuweisung einer Storage-Klasse

Die Zuweisung einer Storage-Klasse an eine Datei (oder Dateigeneration) kann durch explizite Angabe erfolgen (STORAGE-CLASS=<composed-name>) oder über Defaultierung (STORAGE-CLASS=*STD). Eine Storage-Klasse kann einer Datei/-generation erst dann zugewiesen werden, wenn Speicherplatz zugewiesen wird. Zu diesem Zeitpunkt werden auch die ablageort-relevanten Attribute zugewiesen.
Für eine Datei, die keinen Platz zugeordnet hat, sind die Werte der ablageort-relevanten Dateiattribute AVAILABILITY, WORK-FILE und FILE-PREFORMAT sowie die Storage-Klassen-Zuordnung nicht definiert. Dies gilt auch für den Fall, dass einer Datei der zugewiesene Speicherplatz wieder vollständig entzogen wird (Kommandos DELETE-FILE oder MODIFY-FILE-ATTRIBUTES..., SPACE=*RELEASE(...)).

Bei der Rücknahme einer Storage-Klassen-Zuordnung für eine Datei (Kommando MODIFY-FILE-ATTRIBUTES..., STORAGE-CLASS=*NONE) ist Folgendes zu beachten:

Der Name der Storage-Klasse wird aus dem Katalogeintrag ausgetragen. Nur die explizit angegebenen ablageort-relevanten Attribute werden geändert. Um ein Unterlaufen der Kontingentierung zu verhindern, werden aber für Dateien mit dem (resultierenden) Attribut S0-MIGRATION=*FORBIDDEN die Werte für PERFORMANCE und AVAILABILITY ggf. automatisch den entsprechenden Werten des Volume-Sets angepasst, auf dem die Datei liegt.

Defaultierung des Dateipräformats

Die Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES sowie die Kommandos zum Einrichten und Modifizieren von Storage-Klassen erlauben für das Attribut PREFORMAT den Wert *BY-PUBSET- DEFAULT, der auch als Defaultwert dient.
Die Wirkung dieses Defaultwertes hängt davon ab, ob eine Datei durch physikalische Allokierung angelegt wird:

- Bei physikalischer Allokierung erhält eine Datei das Format des Volume-Sets, auf dem sie angelegt wird, als Präformat zugeordnet.
- Eine Datei, die nicht durch physikalische Allokierung angelegt wird, bekommt das Default-File-Format des SM-Pubsets als Präformat zugeordnet. Das Präformat wird bei der Auswahl eines geeigneten Volume-Sets berücksichtigt.

Neben der Volume-Set-Auswahl beeinflusst das Präformat auch die Festlegung des endgültigen Dateiformats einer Datei, wenn diese für Neuerstellung geöffnet wird: Erfolgen dabei beim Öffnen keine expliziten Formatvorgaben durch den Benutzer (ADD-FILE-LINK oder FCB), dient das Präformat als Defaultwert.

Näheres zur Bestimmung des Dateipräformats und des endgültigen Dateiformats siehe die Abschnitte auf "Bestimmung des Präformats" und "Bestimmung des Dateiformats".

Defaultierung der Verfügbarkeit

Wird eine Datei nicht durch physikalische Allokierung angelegt, ist *STD der Defaultwert für Verfügbarkeit. Beim Anlegen einer Datei durch physikalische Allokierung wird der Attributwert wie folgt aus den Benutzerangaben bestimmt:

AVAILABILTY-Wert des Volume-Sets, auf dem die Datei angelegt wird	vom Benutzer spezifiziertes Dateiattribut AVAILABILITY	resultierender Wert des Dateiattributs AVAILABILITY
HIGH	nicht explizit angegeben HIGH STD	HIGH HIGH HIGH
STD	nicht explizit angegeben HIGH STD	STD Angabe unverträglich STD

Wird S0-MIGRATION=*FORBIDDEN gesetzt (explizit über das Kommando MODIFY-FILE-ATTRIBUTES oder implizit durch physikalische Allokierung) und belegt die Datei auf einem Volume-Set mit AVAILABILITY=*HIGH Speicherplatz, so wird für die Datei AVAILABILITY=*HIGH eingestellt (auch wenn AVAILABILITY=*STD explizit angegeben wurde).

Defaultierung der statischen performance-bezogenen Attribute

Bei den Performance-Attributen wird zwischen statischen und dynamischen Werten unterschieden (siehe Abschnitt „Ablauf der OPEN-Verarbeitung“). Statische Werte sind im Katalogeintrag hinterlegt, dynamische werden in der TFT ablgelegt oder im TU-FCB angegeben. Explizite Benutzerangaben bzw. aus den Storage-Klassen entnommene Werte werden ggf. durch Berücksichtigung der DMS-TUNING-RESOURCES-Einstellung für den Dateieigentümer reduziert.

Um beim Anlegen einer Datei durch physikalische Allokierung die Kontingentüberwachung nicht unterlaufen zu können, werden durch CREATE-FILE die Benutzerangaben bzw. bestehende Einstellungen für das statische PERFORMANCE-Attribut abhängig von der Performance-Charakteristik des vorgegebenen Volume-Sets ggf. „angehoben“.

PERFORMANCE-Werte des Volume-Sets	vom Benutzer spezifiziertes Dateiattribut PERFORMANCE	resultierendes Dateiattribut PERFORMANCE bei physikalischer Allokierung
STD [,HIGH] [,VERY-HIGH]	nicht explizit angegeben STD HIGH VERY-HIGH	STD STD HIGH VERY-HIGH
HIGH [,VERY-HIGH]	nicht explizit angegeben STD HIGH VERY-HIGH	HIGH HIGH HIGH VERY-HIGH

VERY-HIGH	nicht explizit angegeben STD HIGH VERY-HIGH	VERY-HIGH VERY-HIGH VERY-HIGH VERY-HIGH

Das „Anheben“ erfolgt auch dann, wenn dadurch die DMS-TUNING-RESOURCES-Berechtigung des Dateieigentümers überschritten wird.

Die Bestimmung des Werts für die statische Datei-Performance durch das Kommando MODIFY-FILE-ATTRIBUTES ist abhängig vom Wert des Attributs S0-MIGRATION nach der Ausführung des Kommandos:

- Ergibt sich durch die Ausführung S0-MIGRATION=*FORBIDDEN, so wird die vom Benutzer spezifizierte Performance (bei expliziter Spezifikation) bzw. die der Datei bisher zugeordnete statische Performance ggf. automatisch modifiziert. Die geschieht nach den gleichen Regeln wie beim Anlegen einer Datei durch physikalische Allokierung. Auch hier kann sich für die statische Performance ein Wert ergeben, der die DMS-TUNING-RESOURCES-Berechtigung des Dateieigentümers überschreitet.
- Ergibt sich durch die Ausführung die Zuweisung S0-MIGRATION=*ALLOWED, so werden explizite Benutzerangaben für die Performance so reduziert, dass die DMS-TUN-ING-RESOURCES-Berechtigung nicht überschritten werden. Erfolgt keine explizite Benutzerangabe für die Performance, wird der bisherige Performance-Wert unverändert beibehalten.

Defaultierung des Dateityps Arbeitsdatei

Per Default werden Dateien als permanente Dateien eingerichtet (WORK-FILE=*NO). Eine Arbeitsdatei kann über folgende Mechanismen eingerichtet werden:

- explizite Zuordnung WORK-FILE=*YES
- physikalische Allokierung (explizite Angabe einer Platte, die zu einem Work-Volume-Set gehört über den VOLUME-Operand bzw. explizite Angabe des Work-Volume-Sets über den Operanden VOLUME-SET)
- Angabe einer Storage-Klasse, in der die Attributzuordnung WORK-FILE=*YES gilt

Für eine Datei, die bereits Speicherplatz belegt, kann die Zuordnung WORK-FILE nicht mehr verändert werden.

Defaultierung des Attributs S0-MIGRATION

Im Kommando CREATE-FILE wird der Defaultwert für dieses Attribut eingestellt.
Er bewirkt im Fall von physikalischer Allokierung die Zuweisung S0-MIGRATION= *FORBIDDEN, andernfalls die Zuweisung S0-MIGRATION=*ALLOWED.

Im Kommando MODIFY-FILE-ATTRIBUTES wird mit S0-MIGRATION=*UNCHANGED der Defaultwert für dieses Attribut eingestellt.
Er bewirkt, dass für eine Datei, die bereits Speicherplatz belegt, der bisherige Wert beibehalten wird, auch wenn sie durch das Kommando gleichzeitig durch physikalische Allokierung erweitert wird.
Für eine Datei, die bisher keinen Speicherplatz belegt und durch MODIFY-FILE-ATTRIBUTES Speicherplatz zugewiesen bekommt, wird der Wert für S0-MIGRATION wie bei dem Neuanlegen einer Datei bestimmt.

Defaultierung des Attributs MIGRATE

Im Kommando CREATE-FILE wird mit MIGRATE=*STD der Defaultwert für dieses Attribut eingestellt.
Er bewirkt im Fall von physikalischer Allokierung auf Volume-oder Blockebene die Zuweisung MIGRATE=FORBIDDEN.
Bei nicht durch physikalische Allokierung angelegten Dateien ist zu unterscheiden, ob es sich um permanente oder temporäre Dateien handelt: bei einer temporären Datei erfolgt die Zuweisung MIGRATE =INHIBITED, bei einer permanenten Dateien erfolgt die Zuweisung MIGRATE=ALLOWED.

Im Kommando MODIFY-FILE-ATTRIBUTES wird mit MIGRATE=*UNCHANGED der Defaultwert für dieses Attribut eingestellt.

Die Wirkung ist davon abhängig, ob die Datei bereits Speicherplatz belegt:

- Für eine Datei, die bereits Speicherplatz belegt, wird der bisherige Wert für MIGRATION beibehalten, falls sie nicht gleichzeitig von einer temporären in eine permanente Datei umgewandelt wird.
- Temporäre Dateien erhalten automatisch die Zuordnung MIGRATE=INHIBITED. Es hat auf die Bestimmung des MIGRATE-Attributwerts keinen Einfluss, ob die Datei durch physikalische Allokierung vergrößert wird.
- Für eine Datei, die bisher keinen Speicherplatz belegt und durch MODIFY-FILE-ATTRIBUTES Speicherplatz zugewiesen bekommt, wird der Wert für MIGRATION wie beim Neuanlegen einer Datei bestimmt.

5.2.3 Ablageortbestimmung bei SF-Pubsets

SF-Pubsets entstehen durch die Zusammenfassung von Volumes, die in ihren Eigenschaften übereinstimmen sollten. Unterschiede zwischen den Volumes werden bei der Speicherplatzbereitstellung für Dateien durch das System nicht berücksichtigt. Die Zuordnung von Caches erfolgt auf der Ebene der Pubsets, nicht auf der Ebene einzelner Volumes. (Homogene) Pubsets bilden damit Behälter für Dateien, die durch ein bestimmtes Eigenschaftsprofil charakterisiert werden können (K- oder NK2-Pubset, DRV-Pubset, ...).

Unterschiedliche Eigenschaftsprofile sind durch eine Menge unterschiedlicher SF-Pubsets zu realisieren. Es obliegt dem Benutzer im Rahmen seiner Berechtigungen, daraus die Pubsets als Ablageort für seine Dateien auszuwählen, die den jeweiligen Anforderungen am besten genügen. Um die Pubset-Auswahl sinnvoll zu treffen, benötigt der Benutzer Kenntnisse über die in einem Rechenzentrum vorhandene Konfiguration.
Informationsmöglichkeiten über die installierten Pubsets und ihre Eigenschaften sind durch folgende Schnittstellen gegeben:

- Das Kommando SHOW-PUBSET-CONFIGURATION gibt in den Informationsblöcken SUMMARY und PHYSICAL-CONFIGURATION Auskunft über den Pubset-Typ (SF bzw. SM), die zum Pubset gehörenden Volumes, ihren Gerätetyp, und die für die Volumes bestehenden Allokierungseinschränkungen (Allokierung uneingeschränkt, nur physikalische Allokierung erlaubt, Allokierung verboten).
- Das Kommando SHOW-MASTER-CATALOG-ENTRY bzw. der Makro STAM zeigen die globalen Eigenschaften eines SF-Pubsets bezüglich Cache-Zuordnung, Verfügbarkeit, Struktur, Allokierungseinheit usw. an.

Nachdem der Benutzer die Auswahl getroffen hat, muss dem System mitgeteilt werden, auf welchem Pubset die Datei liegen soll. Dies geschieht durch den Pfadnamen der Datei, in welchen die Catid des als Ablageort gewünschten Pubsets eingeht. Die Angabe der Catid muss nicht explizit erfolgen, d.h. es können auch Dateinamen ohne Catid-Angabe angegeben werden. Der vollständige Pfadnamen wird in diesem Fall automatisch durch das System unter Heranziehung der benutzer-spezifischen Default-Catid ermittelt.

Bestimmung des Dateiformats

Im Gegensatz zum SM-Pubset ist die Bestimmung eines Präformats nicht erforderlich. Die Ermittlung des Dateiformats, die sich in den Attributen BLOCK-CONTROL-INFO (bzw. BLKCTRL) und BUFFER-LENGTH (bzw. BLKSIZE) widerspiegelt, erfolgt analog.
Die Attribute werden erst während dem Öffnen der Datei (Makro OPEN, Open-Modus OUTPUT bzw. OUTIN) festgelegt, wobei für die Bestimmung entscheidend sind:

- Attributwerte zu FCBTYPE, BLKCTRL, BLKSIZE
- der tatsächliche Ablageort

Die angegebenen Attributwerte zur Bestimmung des Formats werden i.a. übernommen bzw. wie folgt defaultiert:

Plattenformat	Defaultierung von BLKSIZE
K / NK2	(STD,1)
NK4	(STD,2)

Plattenformat	FCBTYPE	Defaultierung von BLKCTRL
K	any	PAMKEY
NK2 / NK4	SAM	DATA
NK2 / NK4	PAM	NO
NK2	ISAM	DATA2K
NK4	ISAM	DATA4K

Bei Angaben, die nicht mit der Zugriffsmethode vereinbar sind, werden während der OPEN-Verarbeitung wie folgt Modifikationen vorgenommen:

FCBTYPE	alter BLKCTRL-Wert	neuer BLKCTRL-Wert
SAM	DATA2K / DATA4K / NO	DATA
PAM	DATA2K / DATA4K	NO
ISAM	DATA / NO	• DATA4K, falls das Plattenformat NK4 ist • DATA2K sonst

Angaben zu BLKCTRL und BLKSIZE, die dem Plattenformat widersprechen, werden abgewiesen, z.B.:

- BLKSIZE=(STD,1) und BLKCTRL=DATA4K
- BLKSIZE=(STD,1) und Datei liegt auf NK4-Platte
- BLKCTRL=DATA4K, BLKSIZE nicht angegeben, FCBTYPE=ISAM und Datei liegt auf K- oder NK2-Platte

5.3 Zugriff auf katalogisierte Dateien

- Zugriff über den Pfadnamen
- Zugriff über die System-Standardkennung
- Zugriff über den Dateikettungsnamen (Link-Name)

5.3.1 Zugriff über den Pfadnamen

Jedem Dateizugriff durch das DVS geht ein Zugriff auf den Katalogeintrag der zu bearbeitenden Datei voraus. Es kann bei der Angabe eines Dateinamens in Makros oder Kommandos in der Regel zwischen verschiedenen Kombinationen von Katalogkennung, Benutzerkennung und Dateinamen gewählt werden; temporäre Dateien werden in der Regel nur mit Präfix und Dateiname angesprochen (Angabe von Standard-Katalogkennung und/oder Standard-Benutzerkennung möglich).

pfadname = [:catid:][$userid.]dateiname

> *dateiname*
> Die Datei wird in dem Katalog gesucht, der der Benutzerkennung des aufrufenden Auftrags zugeordnet ist. Bei einigen Kommandos (bzw. durch die Einstellung OPTION=GLODEF im Makro FCB) wird bei erfolglosem Zugriff die Funktion „Secondary Read“ durchgeführt (siehe folgenden Abschnitt). Ansonsten gibt das DVS eine entsprechende Fehlermeldung aus.

$userid.dateiname

> Der Dateieintrag wird in dem der angegebenen Benutzerkennung standardmäßig zugeordneten Katalog gesucht. Ist die Kennung des aufrufenden Auftrags weder Eigentümer noch Mit-Eigentümer der Datei und auch nicht TSOS, muss die so angesprochene Datei mehrbenutzbar sein. Bei erfolglosem Zugriff reagiert das DVS mit entsprechenden Meldungen.

:catid:$userid.dateiname

> Die Datei wird im Katalog mit der angegebenen Katalogkennung unter der angegebenen Benutzerkennung gesucht. Wird dort kein Eintrag gefunden, gibt das DVS eine entsprechende Meldung aus.

:catid:dateiname

> Die Datei wird in dem so benannten Katalog gesucht, unter der Benutzerkennung des aufrufenden Auftrags. Voraussetzung ist natürlich, dass für die Benutzerkennung in diesem Katalog ein entsprechender Eintrag im Benutzerkatalog existiert.
> Secondary Read ist möglich (siehe folgenden Abschnitt). Bei erfolglosem Zugriff gibt das DVS eine entsprechende Meldung aus.
>
> Im Katalogeintrag ist vermerkt, wie und wo die Datei gespeichert ist (Datenträgerliste und Positionsinformationen).

5.3.2 Zugriff über die System-Standardkennung

Dienstprogramme usw. sind in der Regel unter einer System-Standardkennung katalogisiert. Sollen solche Programme oder Prozeduren genutzt werden, kann der Zugriff auf den Benutzerkatalog umgangen werden, indem diese Standardkennung dem Dateinamen vorangestellt oder die Standardkennung durch Voranstellen des $-Zeichens abgekürzt wird. Dabei muss Folgendes beachtet werden:

- Ist der Dateiname durch Punkte in Teilnamen gegliedert, muss die System-Standardkennung mit „$." abgekürzt werden.
- Ist der Dateiname nicht gegliedert, braucht das $-Zeichen nicht durch einen Punkt vom Dateinamen getrennt zu werden.

Soll auf eine Datei unter der System-Standardkennung zugegriffen werden, und wird dem Dateinamen weder „$." noch „$" vorangestellt, sucht das DVS diese Datei zunächst im Benutzerkatalog der Default-Benutzerkennung des Auftrags. Ist diese Suche erfolglos, führt das DVS bei Kommandos wie START-EXECUTABLE-PROGRAM und CALL-PROCEDURE automatisch den „Secondary Read" durch, einen zweiten Lesezugriff auf den Katalog der System-Standardkennung.

Dieses „Zweite Lesen" kann auch für die OPEN-Verarbeitung mit dem Operanden OPTION=GLODEF im FCB-Makroaufruf vereinbart werden.

Es sollte möglichst wenig Gebrauch von der „Secondary-Read"-Funktion gemacht, sondern die Abkürzungsmöglichkeiten für die System-Standardkennung ($, $.) genutzt werden, da auf diese Weise die Zahl der Katalogzugriffe reduziert werden kann.

Beispiel 1: Makro-Zugriff auf Dateien unter System-Standardkennung / Secondary Read

katalogisierte Dateien:	`$USER1.DAT1`
	`$USER2.DAT1`
	`$DEFL.DAT1`
	`$DEFL.DAT2`
	`$DEFL.DAT.V.1`
Standardkennung:	`$DEFL`

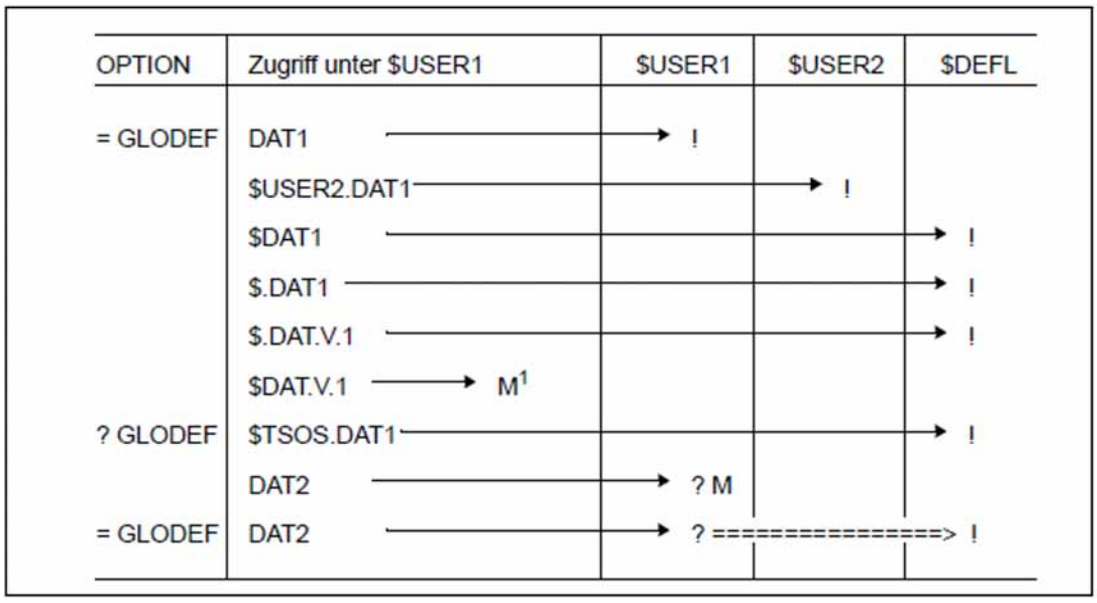


Tabelle 10: Katalogzugriff / Secondary Read (Makro)

Es bedeuten:

`---->`	erster Katalogzugriff
`====>`	zweites Lesen (Secondary Read)
`?`	erfolgloser Katalogzugriff
`!`	Treffer
`M`	Fehlermeldung
`M`[1]	Fehlermeldung, eine Benutzerkennung $DAT. existiert nicht!

Beispiel 2: Kommando-Zugriff auf Dateien unter System-Standardkennung/Secondary Read

katalogisierte Dateien: `$USER1.DAT1`

`$USER2.DAT1`

`$DEFL.DAT1`

`$DEFL.DAT2`

`$DEFL.DAT.V.1`

Standardkennung: `$DEFL`

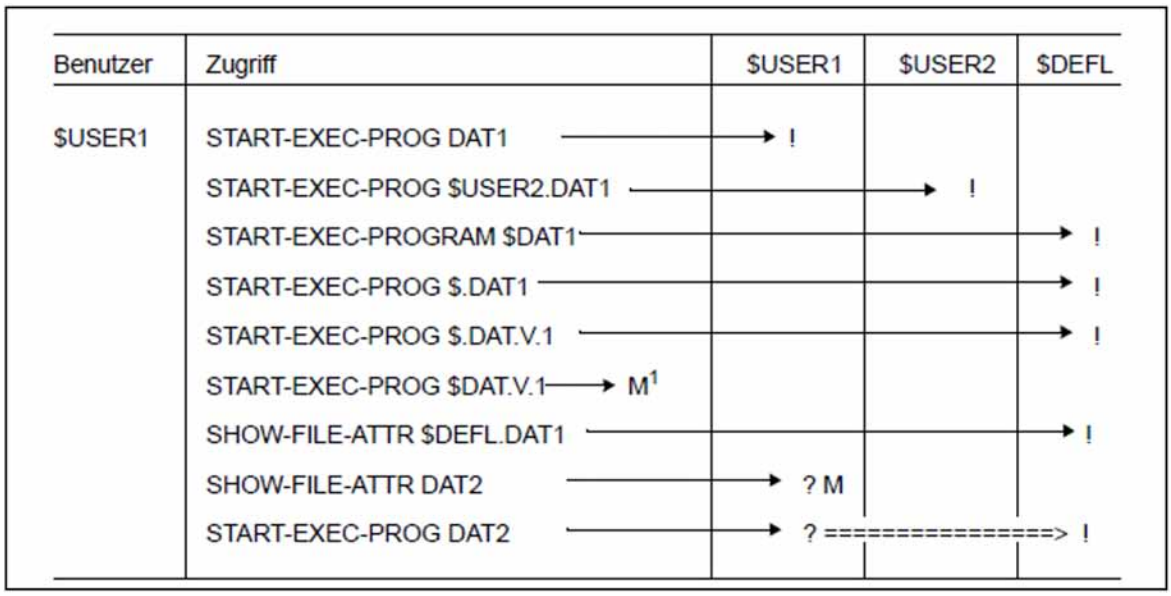


Tabelle 11: Katalogzugriff / Secondary Read (Kommando)

Es bedeuten:

`---->`	erster Katalogzugriff
`====>`	zweites Lesen (Secondary Read)
?	erfolgloser Katalogzugriff
!	Treffer
M	Fehlermeldung
M [1]	Fehlermeldung, eine Benutzerkennung $DAT. existiert nicht!

Beispiel 3: Kommando-Zugriff auf Dateien unter System-Standardkennung/Secondary Read

```
/show-file-attributes file-name=edt ──────────────────────────────  (1)
%       3 :2WR3:$A1234.EDT
%:3WR3: PUBLIC:     1 FILE  RES=         3 FRE=        2 REL=       0 PAGES
/start-exec-program from-file=edt  ──────────────────────────────  (2)
%  BLS0518 INVALID FORMAT OF LOAD MODULE. COMMAND REJECTED
%  NRTT101 ABNORMAL JOBSTEP TERMINATION BLS0518
/start-exec-program from-file=$edt ──────────────────────────────  (3)
%  BLS0500 PROGRAM 'EDT', VERSION '17.0A10' OF '2009-03-06' LOADED
%  BLS0552 COPYRIGHT (C) FUJITSU TECHNOLOGY SOLUTIONS   2009. ALL RIGHTS
RESERVED
:
%  EDT8000 EDT TERMINATED
/show-file-attributes $edt ──────────────────────────────────────  (4)
%       6 :3WWW:$TSOS.EDT
%:3WWW: PUBLIC:     1 FILE  RES=         6 FRE=        2 REL=       0 PAGES
```

(1) Für die Benutzerkennung $A1234 ist eine Datei „EDT“ katalogisiert.

(2) Die Datei „EDT“ enthält kein ablauffähiges Programm, das Kommando START-EXECUTABLE-PROGRAM wird zurückgewiesen.

(3) Es wird der Dateiaufbereiter $EDT aufgerufen, der unter der System-Standardkennung katalogisiert ist.

(4) Das Kommando SHOW-FILE-ATTRIBUTES zeigt den Katalogeintrag des Datei aufbereiters $EDT.

5.3.3 Zugriff über den Dateikettungsnamen (Link-Name)

Soll ein Programm Dateien verarbeiten, muss das DVS eine Verbindung zwischen Programm und Datei herstellen können. Dies geschieht entweder über einen im Programm fest verankerten Dateinamen oder über einen programminternen Dateikettungsnamen, der vor Dateieröffnung einer Datei zuzuordnen ist. Der Zugriff über einen im Programm fest verankerten Dateinamen verläuft wie auf "Zugriff über den Pfadnamen" beschrieben, hat jedoch einige Nachteile:
So ist das Programm z.B. nicht mehr flexibel, bei Änderung des Dateinamens muss das Quellprogramm geändert werden. So erzeugte Ausgabedateien müssen umbenannt/umkatalogisiert werden, damit sie beim nächsten Programmlauf nicht überschrieben werden. Werden Eingabedateien derart verankert, kann das Programm immer nur mit einer einzigen Eingabedatei arbeiten (wenn nicht mehrfach umkatalogisiert wird). Außerdem muss der Programmierer bereits den endgültigen Dateinamen kennen, was z.B. dadurch erschwert wird, dass ein Programm oft für verschiedene Benutzer bzw. Anwendungen geschrieben wird.

Datei und Programm sollten über einen im Programm intern geführten Namen, den Dateikettungsnamen verknüpft werden. Die Zuordnung von Datei zu Dateikettungsname (Link-Name) erfolgt über einen Eintrag in der auftragsbezogenen TFT (Task File Table), der mit dem Makro FILE und dem Operanden LINK bzw. dem Kommando ADD-FILE-LINK und dem Operanden LINK-NAME erzeugt wird. Diese Zuordnung kann immer, wenn der TFT-Eintrag inaktiv ist, aufgehoben (Makro RELTFT oder Kommando REMOVE-FILE-LINK) oder geändert werden (Makros FILE und CHNGE oder Kommandos CHANGE-FILE-LINK und ADD-FILE-LINK).

TFT-Einträge können mit dem Kommando LOCK-FILE-LINK gegen Löschen geschützt werden, d.h. ein REMOVE-FILE-LINK-Kommando oder RELTFT-Makro wird ausgesetzt. Solche gesperrten TFT-Einträge können mit dem Makro DROPTFT oder dem Kommando UNLOCK-FILE-LINK freigegeben werden. Steht für diesen Eintrag noch ein REMOVE-FILE-LINK-Kommando bzw. ein RELTFT-Makro an, wird dieser jetzt bearbeitet, d.h. der TFT-Eintrag wird entsprechend den dort gemachten Angaben gelöscht und die mit ihm verbundenen privaten Geräte werden freigegeben.

Die Felder des TFT-Eintrags beschreiben Datei- und Verarbeitungsmerkmale für die aktuelle Dateiverarbeitung. Es werden alle Felder versorgt, deren zugehörige Operanden in den Makros bzw. Kommandos angegeben werden, die den TFT-Eintrag erstellen oder die sich auf den TFT-Eintrag beziehen.

Bei Dateieröffnung haben die Angaben im TFT-Eintrag Vorrang vor den entsprechenden Feldern des Dateisteuerblocks, der über den FCB-Makroaufruf erstellt wurde.
Eine Sonderstellung haben dabei die Felder des TFT-Eintrags, die mit dem Wert *BY-CAT bzw. *BY-CATALOG versorgt wurden (sog. „Null-Operanden“, siehe unten).

Für diejenigen Eigenschaften, die weder im TFT-Eintrag noch im Dateisteuerblock vereinbart wurden, übernimmt das DVS Standardwerte.

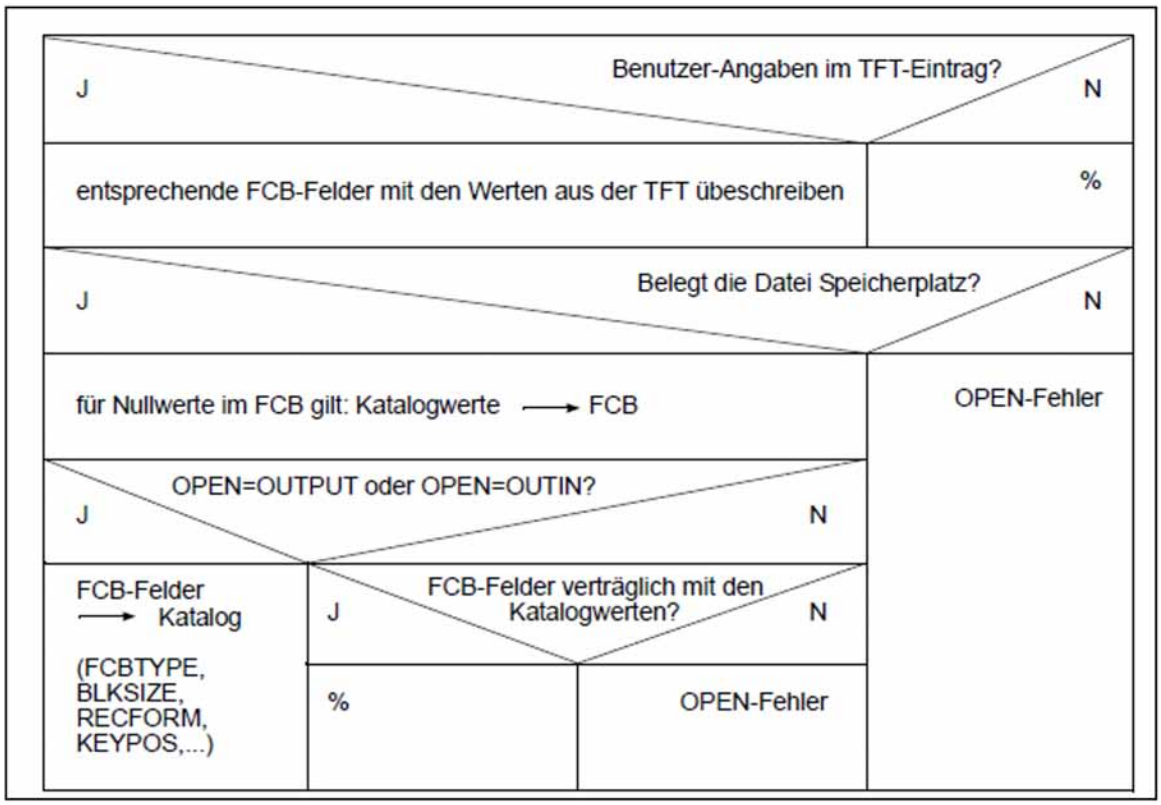


Bild 4: Sammeln von Dateiattributen

Über den Aufbau und den Inhalt der TFT-Einträge kann man sich jederzeit mit dem Makro RDTFT bzw. mit dem Kommando SHOW-FILE-LINK informieren. Mit der Standardfunktion des Makros bzw. des Kommandos wird Auskunft darüber erteilt, welche TFT-Einträge für den laufenden Auftrag erzeugt wurden. Es können jedoch auch gezielt Informationen zu bestimmten TFT-Einträgen abgerufen werden.

Die TFT enthält die Einträge in der Reihenfolge, wie sie durch Makro- oder Kommando-Aufrufe erstellt wurden. Nicht mehr benötigte TFT-Einträge sollten gelöscht werden, da bei jedem Zugriff auf die TFT die Tabelle sequenziell durchsucht wird, sodass sich die Zahl der TFT-Einträge auf die Performance auswirkt.

> ¡ Software-Produkte wie Dienstprogramme, Dateiaufbereiter usw. verwenden oft eigene Dateikettungsnamen, die den jeweiligen Beschreibungen zu entnehmen sind.

Null-Operanden

Bei der Verarbeitung existenter Dateien kann die Null-Operanden-Funktion genutzt werden. Damit wird sichergestellt, dass der Dateisteuerblock zum OPEN-Zeitpunkt korrekt versorgt ist. Eine Reihe von Operanden der Makros FILE und FCB bzw. des Kommandos ADD-FILE-LINK, die Dateieigenschaften beschreiben, können als Null-Operanden im Makroaufruf (leere Zeichenfolge als Operandenwert) bzw. mit dem Operandenwert *BY-CATALOG im Kommandoaufruf angegeben werden. Das entsprechende Feld des TFT-Eintrags enthält dann den Wert „BY-CAT“. Bei Dateieröffnung übernimmt das DVS die Werte aus dem Katalogeintrag in den Dateisteuerblock.

Werden mit Kommandos oder Makros für eine bereits bestehende Datei bestimmte Eigenschaften angegeben, so müssen diese Angaben mit denen im Katalog verträglich sein. Verträglichkeit zwischen FCB und Katalog bedeutet nicht unbedingt Gleichheit der jeweiligen Werte. So können z.B. SAM-/ISAM-Dateien mit der Zugriffsmethode UPAM bearbeitet werden (Zugriffsmethode PAM im FCB).

Beispiel 1: Dateikettungsname – Sortieren mit dem Dienstprogramm SORT

Der Inhalt der Datei DATEN.UNSORT soll mithilfe des Dienstprogramms SORT sortiert werden; die Ausgabedatei erhält den Namen DATEN.SORT. Diese Datei soll durch das Programm DRUCK als druckaufbereitete Liste mit dem Namen LISTE ausgegeben werden. Das Dienstprogramm SORT verwendet standardmäßig die Dateikettungsnamen SORTIN und SORTOUT; das Programm DRUCK verwendet die Dateikettungsnamen LISTEIN und LISTAUS.

```
/show-file-attributes daten.unsort  ————————————————————————————  (1)
%      12 :2OS2:$USER1.DATEN.UNSORT
%:2OS2: PUBLIC:      1 FILE  RES=       12 FRE=        7 REL=       6 PAGES
/add-file-link link-name=sortin,file-name=daten.unsort ——————————————  (2)
/add-file-link link-name=sortout,file-name=daten.sort ———————————————  (3)
/show-file-link   ———————————————————————————————————————————————  (4)
%
%-- LINK-NAME --------- FILE-NAME ----------------------------------------
%   SORTIN              :2OS2:$USER1.DATEN.UNSORT
%   SORTOUT             :2OS2:$USER1.DATEN.SORT
/start-sort  ————————————————————————————————————————————————————  (5)
%  BLS0523 ELEMENT 'SRT80', VERSION '079', TYPE 'L' FROM LIBRARY
':1OSH:$TSOS.SYSLNK.SORT.080' IN PROCESS
%  BLS0524 LLM 'SRT80', VERSION '08.0A00' OF '2015-02-03 13:07:17' LOADED
%  BLS0551 COPYRIGHT (C) 2014 FUJITSU TECHNOLOGY SOLUTIONS GMBH. ALL RIGHTS
RESERVED
%  SRT1001  2017-03-03/18:19:43/000000.00 SORT/MERGE STARTED, VERSION
08.0A00/BS2000V20.0
%  SRT1130  PLEASE ENTER SORT STATEMENTS
*sort fields=(10,5,a,ch)
*record length=100
*end
%  SRT1222  WARNING: RECORD SIZE OF INPUT FILE NOT EQUAL TO <LENGTH1>.
            <LENGTH1> WILL BE IGNORED
%  SRT1016  SORT/MERGE INPUT RECORDS:.........................101 (FROM 01)
%  SRT1030  SORT/MERGE OUTPUT RECORDS:.........................101
%  SRT1002  09:49:11/000000.16 SORT/MERGE COMPLETED
/show-file-link ————————————————————————————————————————————————  (6)
%
%-- LINK-NAME --------- FILE-NAME ----------------------------------------
%   SORTOUT             :2OS2:$USER1.DATEN.SORT
/change-file-link link-name=sortout,new-name=listein ———————————————  (7)
/add-file-link link-name=listaus,file-name=liste   ——————————————————  (8)
/show-file-link   ———————————————————————————————————————————————  (9)
%
%-- LINK-NAME --------- FILE-NAME ----------------------------------------
%   LISTAUS             :2OS2:$USER1.LISTE
%   LISTEIN             :2OS2:$USER1.DATEN.SORT
/start-prog from-file=druck ————————————————————————————————————— (10)
%  BLS0517 MODULE 'DRUCK' LOADED
:
: *** Erstellen der Ausgabeliste im Programm DRUCK ***
:
*** PROGRAMM DRUCK BEENDET ***
/remove-file-link link-name=listein ————————————————————————————— (11)
/remove-file-link link-name=listaus ————————————————————————————— (12)
/show-file-link ————————————————————————————————————————————————— (13)
%  DMS05E1 TASK FILE TABLE (TFT) NOT AVAILABLE OR SPECIFIED FILE NOT IN
           'TFT'. OPERATION NOT PROCESSED
```

(1) In der Datei „DATEN.UNSORT“ befinden sich Daten.

(2) Eingabedatei mit dem Dienstprogramm SORT verbinden

(3) Ausgabedatei mit dem Dienstprogramm SORT verbinden

(4) TFT-Einträge anzeigen

(5) Aufruf des Dienstprogramms SORT

`"*"` Eingabe der Sortieranweisungen

`"% SRT..."` Meldungen des Dienstprogrammes SORT

(6) TFT-Einträge anzeigen: der TFT-Eintrag für die SORT-Eingabedatei SORTIN wurde vom Dienstprogramm SORT freigegeben

(7) Umbenennen von Link-Namen: die Ausgabedatei für das Dienstprogramm SORT (SORTOUT) wird umbenannt in die Eingabedatei für das Druckprogramm DRUCK (LISTEIN)

(8) Für die DRUCK-Ausgabedatei wird ein TFT-Eintrag erstellt

(9) TFT-Einträge anzeigen

(10) Das Programm DRUCK wird gestartet

(11) TFT-Eintrag für die DRUCK-Eingabedatei (LISTEIN) löschen

(12) TFT-Eintrag für die DRUCK-Ausgabedatei (LISTAUS) löschen

(13) Das Kommando SHOW-FILE-LINK zeigt, dass alle TFT-Einträge für diesen Auftrag gelöscht sind

Beispiel 2: Übernahme des gültigen Wertes aus Katalogeintrag

```
/sh-f-attr lst.bsp.2,inf=par(org=yes)
%0000000003 :2OS2:$USER1.LST.BSP.2
% ----------------- ORGANIZATION ----------------------------------------
% FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
% IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
% REC-FORM   = (V,M)       REC-SIZE   = 0
% AVAIL      = *STD
% WORK-FILE  = *NO         F-PREFORM  = *K          S0-MIGR    = *ALLOWED
%:2OS2: PUBLIC:      1 FILE  RES=        3 FRE=         2 REL=        0 PAGES
/add-file-link link=edtsam,file-name=lst.bsp.2,access-method=*by-cat,
rec-form=*by-cat,buffer-length=*by-cat,block-contr-info=*by-cat
/show-file-link link=edtsam,inf=all
%-- LINK-NAME --------- FILE-NAME ----------------------------------------
%   EDTSAM              :2OS2:$USER1.LST.BSP.2
%  ------------------- STATUS ----------------------------------------
%  STATE      = INACTIVE    ORIGIN     = FILE
%  ------------------- PROTECTION ------------------------------------
%  RET-PER    = *BY-PROG    PROT-LEV   = *BY-PROG
%  BYPASS     = *BY-PROG    DESTROY    = *BY-CAT
%  ------------------- FILE-CONTROL-BLOCK - GENERAL ATTRIBUTES ------------
%  ACC-METH   = *BY-CAT     OPEN-MODE  = *BY-PROG    REC-FORM   = *BY-CAT
%  REC-SIZE   = *BY-PROG    BUF-LEN    = *BY-CAT     BLK-CONTR  = *BY-CAT
%  F-CL-MSG   = STD         CLOSE-MODE = *BY-PROG
%  ------------------- FILE-CONTROL-BLOCK - DISK FILE ATTRIBUTES ----------
%  SHARED-UPD = *BY-PROG    WR-CHECK   = *BY-PROG    IO(PERF)   = *BY-PROG
%  IO(USAGE)  = *BY-PROG    LOCK-ENV   = *BY-PROG
%  ------------------- FILE-CONTROL-BLOCK - TAPE FILE ATTRIBUTES ----------
%  LABEL      = *BY-PROG   (DIN-R-NUM  = *BY-PROG,   TAPE-MARK  = *BY-PROG)
%  CODE       = *BY-PROG    EBCDIC-TR  = *BY-PROG    F-SEQ      = *BY-PROG
%  CP-AT-BLIM = *BY-PROG    CP-AT-FEOV = *BY-PROG    BLOCK-LIM  = *BY-PROG
%  REST-USAGE = *BY-PROG    BLOCK-OFF  = *BY-PROG    TAPE-WRITE = *BY-PROG
%  STREAM     = *BY-PROG
%  ------------------- FILE-CONTROL-BLOCK - ISAM FILE ATTRIBUTES ----------
%  KEY-POS    = *BY-PROG    KEY-LEN    = *BY-PROG    POOL-LINK  = *BY-PROG
%  LOGIC-FLAG = *BY-PROG    VAL-FLAG   = *BY-PROG    PROPA-VAL  = *BY-PROG
%  DUP-KEY    = *BY-PROG    PAD-FACT   = *BY-PROG    READ-I-ADV = *BY-PROG
%  WR-IMMED   = *BY-PROG    POOL-SIZE  = *BY-PROG
%  ------------------- VOLUME ----------------------------------------
%  DEV-TYPE   = *NONE       T-SET-NAME = *NONE
%  VSN/DEV    = GVS2.5/D3435
```

Die Operanden ACCESS-METHOD, RECORD-FORMAT, BUFFER-LENGTH und BLOCK-CONTR-INFO wurden im Kommando ADD-FILE-LINK mit „BY-CATALOG“ angegeben, da der gültige Wert bei Dateieröffnung aus dem Katalogeintrag genommen werden soll. Sie werden bei der Verarbeitung über den Kettungsnamen *EDTSAM* aus dem Katalogeintrag der Datei *LST.BSP.2* übernommen. Der TFT-Eintrag zeigt in den entsprechenden Feldern den Wert „*BY-CAT“.

5.4 Zugriff auf nicht-katalogisierte Dateien

Nicht-katalogisierte Dateien sind immer Dateien auf privaten Datenträgern oder Net-Storage-Volumes, die exportiert oder in anderen Rechenzentren erstellt wurden, Banddateien evtl. sogar mit anderen Betriebssystemen.

5.4.1 Nicht-katalogisierte Plattendateien

Sollen auf Privatplatten oder Net-Storage-Volumes gespeicherte nicht-katalogisierte Dateien verarbeitet werden, können diese mit den Makros IMPORT oder FILE (Operand STATE=FOREIGN) bzw. mit dem Kommando IMPORT-FILE importiert werden. Es muss dann jeweils auch der Datenträger benannt werden, auf dem die Datei beginnt, damit das DVS aus dem F1-Kennsatz der Privatplatte oder dem Katalog des Net-Storage-Volumes den Katalogeintrag der Datei erstellen kann.

Beim Importieren können mehrere Dateien auf einmal importiert werden oder sogar alle Dateien eines Benutzers, die auf dem im Makro- oder Kommandoaufruf benannten Datenträger beginnen. Der Katalogeintrag wird automatisch aus dem F1-Kennsatz der Platte oder dem Katalog des Net-Storage-Volumes erstellt.

Handelt es sich bei den zu importierenden Dateien um Dateigenerationen, muss vor dem Importieren der Gruppeneintrag erstellt werden: entweder muss als Erstes die Platte importiert werden, die den Gruppeneintrag enthält, oder es muss ein Gruppeneintrag mit dem Makro CATAL (Operand STATE=FOREIGN) bzw. mit dem Kommando CREATE-FILE-GROUP (Operand VOLUME) erstellt werden (Näheres siehe Abschnitt „Dateigenerationen / Dateigenerationsgruppen importieren“).

Dateien, die von offenen Systemen in einem benutzerspezifischen Verzeichnis eines Net-Storage-Volumes abgelegt wurden, können als Node-Files mit dem Makro IMPNFIL bzw. dem Kommando IMPORT-NODE-FILE in den Katalog des Net-Storage-Volumes sowie in den TSOSCAT des zugehörigen Pubsets aufgenommen werden. Die Dateien müssen den Dateinamenskonventionen des BS2000 entsprechen. Der Benutzer kann dabei entscheiden, ob die Datei als PAM- oder SAM-Datei katalogisiert werden soll. PAM-Dateien werden mit BLKSIZE=(STD,1) und SAM-Dateien mit BLKSIZE=(STD,16) katalogisiert. Die Zeichensätze CCS und NETCCS werden entsprechend der Vorgaben aus dem Benutzereintrag vorbelegt.

5.4.2 Nicht-katalogisierte Banddateien

Soll eine nicht-katalogisierte Banddatei verarbeitet werden, muss die Datei importiert und die ihr zugeordneten Datenträger beim Makro- oder Kommandoaufruf exakt beschrieben werden.

Wenn die Datei mit Standard-Kennsätzen erstellt wurde, kann die Angabe von Dateiattributen wie RECFORM (RECORD-FORMAT), RECSIZE (RECORD-SIZE), BLKSIZE (BUFFER-LENGTH), CODE usw. entfallen, da diese Merkmale dem HDR2-Kennsatz entnommen werden.

Bei kennsatzlosen Dateien, bzw. bei Dateien mit Nicht-Standard-Kennsätzen, müssen alle Dateieigenschaften explizit mit FILE (Makro) bzw. mit ADD-FILE-LINK (Kommando) angegeben werden, da andernfalls Standardwerte eingesetzt werden.

Für Dateien mit Standard-Kennsätzen ist voller Zugriffsschutz gewährleistet.

Die folgenden Tabellen geben einen Überblick über die Operanden des Makros FILE bzw. des Kommandos ADD-FILE-LINK, die beim Zugriff auf nicht-katalogisierte Banddateien zu verwenden sind:

Makro FILE

Operand	**Kennsatz-Angabe**		
	LABEL=(STD,n) Standard-Kennsätze	**LABEL=NSTD Nicht-Standard-Kennsätze**	**LABEL=NO keine Kennsätze**
dateiname	+	+	+
LINK	(1)	(1)	(1)
DEVICE	+	+	+
VOLUME	+	+	+
MOUNT	x	x	x
STATE=FOREIGN	+	+	+
SECLEV	x	x	x
FCBTYPE	x	x	x
BLKCTRL	(2)	(2)	(2)
RECFORM	x	+	+
RECSIZE	x	+	+
BLKSIZE	x	+	+
BUFOFF	x	+	+
LABEL	x	+	+
TPMARK		x	x
FSEQ	(3)		(3)

CODE	x	+	+

Tabelle 12: Operanden des Makros FILE für nicht-katalogisierte Banddateien

+ der Operand muss angegeben werden

x der Operand kann angegeben werden

(1) muss angegeben werden, damit ein TFT-Eintrag angelegt wird und Dateieigenschaften wie BLKSIZE, RECFORM usw. ausgewertet werden können

(2) muss angegeben werden, wenn die Datei eine auf Band kopierte NK-ISAM-Datei ist

(3) muss nicht angegeben werden, wenn die Datei die erste Datei einer Dateimenge bzw. eines MF/MV-Sets ist

Kommando ADD-FILE-LINK

Operand	**Kennsatz-Angabe**		
	LABEL=(STD,n) Standard-Kennsätze	**LABEL=NSTD Nicht-Standard-Kennsätze**	**LABEL=NO keine Kennsätze**
FILE-NAME	+	+	+
LINK-NAME	(1)	(1)	(1)
ADD-CATALOG-VOLUME	+	+	+
NUMBER-OF-PREMOUNTS	x	x	x
PROTECTION-LEVEL	x	x	x
ACCESS-METHOD	x	x	x
BLOCK-CONTROL-INFO	(2)	(2)	(2)
RECORD-FORMAT	x	+	+
RECORD-SIZE	x	+	+
BUFFER-LENGTH	x	+	+
BLOCK-OFFSET	x	+	+
LABEL	x	+	+
TAPE-MARK		x	x
FILE-SEQUENCE	(3)		(3)
CODE	x	+	+

Tabelle 13: Operanden des Kommandos ADD-FILE-LINK für nicht-katalogisierte Banddateien

+ der Operand muss angegeben werden

x der Operand kann angegeben werden

(1) muss angegeben werden, damit ein TFT-Eintrag angelegt wird und Dateieigenschaften wie BUFFER-LENGTH, RECORD-FORMAT usw. ausgewertet werden können

(2) muss angegeben werden, wenn die Datei eine auf Band kopierte NK-ISAM-Datei ist

(3) muss nicht angegeben werden, wenn die Datei die erste Datei einer Dateimenge bzw. eines MF/MV-Sets ist

5.4.3 Nicht-BS2000-Banddateien

Banddateien, die nicht im BS2000 erstellt wurden, werden genauso behandelt wie nichtkatalogisierte BS2000-Dateien. Zugriffsschutz ist jedoch auf Grund der oft unterschiedlichen Betriebssystem-Prinzipien nicht immer in vollem Umfang möglich. Bei Datenträgern und Dateien mit Standardkennsätzen nach DIN 66029 wird Eigentümerschutz gewährleistet, falls ein Eigentümerkennzeichen im VOL1-Kennsatz eingetragen ist und weder der Datenträger noch die Dateien mehrbenutzbar sind.

5.5 ACS: Dateizugriff über ein Alias-Katalogsystem

Mit ACS (Alias Catalog Service) ist es möglich, auf Dateien und JVs unter zusätzlichen, in gewissen Grenzen frei wählbaren Namen zuzugreifen. Der Benutzer hat damit die Möglichkeit, Aliasnamen für die von ihm benötigten Dateien/JVs zu definieren und zusammen mit der Zuordnung zur realen Datei/JV in speziellen Katalogen, den Aliaskatalogen, zu hinterlegen. Bei der Bearbeitung der Datei/JV ist dann nur noch der Aliasname anzugeben. Die Kataloge werden tasklokal geführt.

Überblick über die ACS-Funktionen und deren Vorteile

Der Alias Catalog Service (ACS) umfasst drei Grundfunktionen:

- Aliasnamen-Vereinbarung
- Catid-Einfügung für temporäre Spooldateien
- Präfix-Einfügung

Funktion Aliasnamen-Vereinbarung

Innerhalb einer Task können durch ACS-Kommandos Aliasnamen für Dateien/JVs definiert werden. Die Definitionen werden im so genannten (temporären) Aliaskatalog hinterlegt und können in permanenten Dateien abgespeichert und bei Bedarf wieder in den Aliaskatalog (auch von anderen Tasks) geladen werden. Beim Zugriff auf eine Datei/JV wird der Aliasname durch den realen Namen ersetzt. Die Alias-Vereinbarungen gelten tasklokal. Der Aliaskatalog wird bei Taskende gelöscht.

Mit dieser Funktion können Softwareprodukte unabhängig von Versionsangaben oder RZ-Kennungen angesprochen werden. Die dazu nötige Information muss allerdings von der Systembetreuung in Form einer Aliaskatalog-Systemdatei zur Verfügung gestellt werden. Der Benutzer kann damit unabhängig von RZ-Konventionen bei der Festschreibung der Namen von Dateien/JVs, der Benutzer- und Katalogkennungen sowie unabhängig von Versionsumstellungen arbeiten; seine Prozeduren bleiben portabel.
Dateien/JVs, die zu entkoppelten Softwareprodukten gehören, können unabhängig von Versionen und RZ-Kennungen mit immer denselben Aliasnamen in einem Aliaskatalog angesprochen werden.

Funktion Catid-Einfügung für temporäre Spooldateien

Systemweit kann eine Katalogkennung (Catid) für die von der Spooltask erzeugten temporären Dateien/JVs festgelegt werden. Diese Dateien/JVs werden dann nicht mehr auf dem Default-Pubset der Nutzertasks abgelegt. Diese Funktion ist nur für den ACS-Administrator relevant.

Diese Funktion zeigt sich für den nicht-privilegierten Benutzer nur in den Ausgabefeldern des Kommandos SHOW-FILE-ATTRIBUTES.

Funktion Präfix-Einfügung

Innerhalb einer Task kann ein Präfix definiert werden, das bei Eingabe von Datei-/JV-Namen, entsprechend den Regeln, jeweils implizit davor gesetzt wird.

Die Funktion Präfix-Einfügung bietet eine komfortable Möglichkeit zur Strukturierung der Namen von Dateien/JVs innerhalb einer Benutzerkennung. Dateien/JVs können nach funktionalen Gesichtspunkten zusammengefasst werden. Der Benutzer kann sich so innerhalb seiner Benutzerkennung eine Art Unterkatalog einrichten, die er als lokale Arbeitsumgebung nutzen kann. Auf einer gemeinsamen Benutzerkennung können Dateien/JVs unter einem namens-, produkt- oder versionsbezeichnenden Präfix (Namen, Produkt- und Versionsbezeichnung) angesprochen werden.

5.5.1 Kommando- und Ablaufübersicht für ACS

Kommando	Bedeutung
ADD-ALIAS-CATALOG-ENTRY	fügt einen Eintrag im Aliaskatalog der aktuellen Task hinzu und legt den Wirkungsbereich der Ersetzung fest
HOLD-ALIAS-SUBSTITUTION	unterbricht die ACS-Funktion für die laufende Task; es findet keine Namensersetzung mehr statt
LOAD-ALIAS-CATALOG	übernimmt gespeicherte Einträge aus einer AC-Datei in den Aliaskatalog
MODIFY-ACS-OPTIONS	ändert tasklokale Optionen
MODIFY-ALIAS-CATALOG-ENTRY	ändert bestehenden Eintrag im Aliaskatalog
PURGE-ALIAS-CATALOG	löscht den Alias-Katalog der laufenden Task; die eingestellten Optionen bleiben jedoch erhalten
REMOVE-ALIAS-CATALOG-ENTRY	löscht bestehenden Eintrag im Aliaskatalog
RESUME-ALIAS-SUBSTITUTION	nimmt die ACS-Funktion Namensersetzung wieder auf
SET-FILE-NAME-PREFIX	vereinbart Präfix für Datei-/JV-Namen und legt Wirkungsbereich des Präfix fest
SHOW-ACS-OPTIONS	informiert über tasklokale Optionen
SHOW-ACS-SYSTEM-FILES	informiert über vordefinierte Aliaskatalog-Systemdateien
SHOW-ALIAS-CATALOG-ENTRY	informiert über ausgewählte Aliaskatalog-Einträge (SYSOUT)
SHOW-FILE-NAME-PREFIX	zeigt aktuellen Präfix und seinen Gültigkeitsbereich an
STORE-ALIAS-CATALOG	speichert Aliaskatalog in Datei ab

Tabelle 14: Kommandoübersicht ACS (Benutzerkommandos)

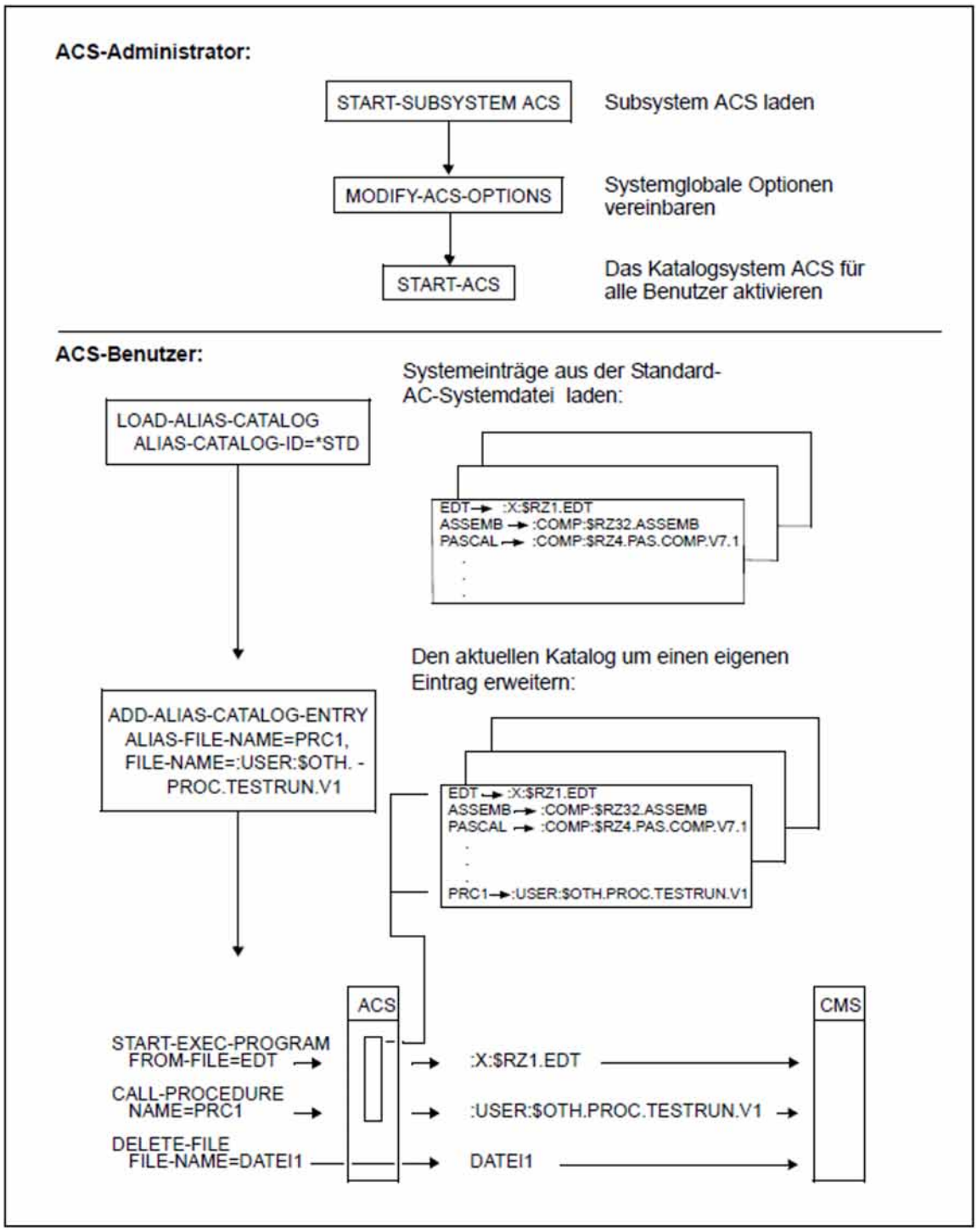


Bild 5: Funktion von ACS am Beispiel „Aliasnamen-Vereinbarung“

5.5.2 Der ACS-Administrator

Der ACS-Administrator ist Inhaber des Systemprivilegs ACS-ADMINISTRATION (steht SECOS nicht zur Verfügung, ist das Privileg an die Benutzerkennung TSOS gebunden).
Der ACS-Administrator verfügt über die folgenden, systemweit wirkenden ACS-Optionen:

- Start und Parametrisierung des AC-Systems (d.h. Vergabe von systemglobalen Voreinstellungen)
- Vergabe bzw. Einschränkung der Berechtigungen
- Bereitstellung und Pflege der AC-Systemdateien

Zur Realisierung dieser Aufgaben stehen dem ACS-Administrator spezielle Erweiterungen der ACS-Benutzerkommandos sowie zusätzliche Administrator-Kommandos zur Verfügung.

5.5.3 Aliasnamen-Vereinbarung

Die Funktion „Aliasnamen-Vereinbarung“ des Alias Catalog Service ermöglicht den Zugriff auf Dateien/JVs unter zusätzlichen, in gewissen Grenzen frei wählbaren Namen, den so genannten „Aliasnamen“. Die Dateien/JVs bleiben weiterhin unter ihren Originalnamen ansprechbar. Für eine Datei/JV können beliebig viele Aliasnamen vergeben werden.

Jeder Name einer Datei/JV, der dem DVS per Kommando oder Makro übergeben wird, wird dahingehend überprüft, ob er im Kontext der aktuellen Task als Aliasname definiert ist.
Ist der angegebene Name als Aliasname definiert, wird er durch den korrespondierenden realen Namen ersetzt, und anschließend, wenn nötig, um Katalog- und Benutzerkennung komplettiert. Er bezeichnet dann als vollständiger Pfadname eindeutig eine Datei/JV. Ist der angegebene Name nicht als Aliasname definiert, wird er als realer Name interpretiert.

Aliaskatalog (AC)

Die Definition eines Aliasnamens, der an Stelle des realen Namens einer Datei/JV verwendet werden kann, wird im so genannten Aliaskatalog hinterlegt. Dieser enthält die Zuordnungen von Alias- zu realen Namen. Jede Task hat ihren eigenen Aliaskatalog, der nur durch spezielle ACS-Kommandos aufgebaut, erweitert oder verändert werden kann (siehe Kommandoübersicht auf "Kommando- und Ablaufübersicht für ACS").

Ein Eintrag im Aliaskatalog besteht aus dem Aliasnamen, dem korrespondierenden realen Namen, einem Indikator RANGE für den Gültigkeitsbereich sowie einigen speziellen Attributen. Im Aliaskatalog einer Task können Benutzereinträge und Systemeinträge enthalten sein.

Der Operand RANGE im Kommando ADD-ALIAS-CATALOG-ENTRY bestimmt den Gültigkeitsbereich der Aliasnamen-Ersetzung.

Folgende Einstellungen sind möglich:

*STD	Die Aliasnamen-Ersetzung erfolgt mit dem Wert der für die Task gültigen ACS-Option STANDARD RANGE (Voreinstellung ist *BOTH).
*FILE	Die Aliasnamen-Ersetzung gilt nur für Dateien.
*JV	Die Aliasnamen-Ersetzung gilt nur für JVs.
*BOTH	Voreinstellung: Die Aliasnamen-Ersetzung gilt für Dateien und JVs.

Da ein Eintrag im Aliaskatalog genau über den Aliasnamen identifiziert wird, kann es für einen bestimmten Aliasnamen nur genau einen Eintrag geben. Eine frühere Vereinbarung für einen Aliasnamen wird durch eine neue Vereinbarung für den gleichen Aliasnamen abgelöst, auch wenn die beiden Vereinbarungen unterschiedliche Gültigkeitsbereiche haben.

Jede Task hat ihren eigenen Aliaskatalog, der nur durch ACS-Kommandos mit Einträgen versorgt werden kann. Er wird beim ersten LOAD- oder ADD-ALIAS-CATALOG-ENTRY-Kommando angelegt. Bei Taskende wird der Aliaskatalog gelöscht.

Aliaskatalogdatei (AC-Datei)

Die Einträge eines Aliaskatalogs können in einer Datei abgespeichert werden.
Nur der ACS-Administrator kann auch Systemeinträge in einer Datei abspeichern.
Eine solche Datei wird als Aliaskatalogdatei (AC-Datei) bezeichnet. Die Aliaskatalogdatei ist eine SAM-Datei und enthält die Einträge in einer internen Darstellung.

Die Einträge können aus der Datei mit dem entsprechenden Kommando wieder in den Aliaskatalog zurückgeladen werden. Der nicht-privilegierte Benutzer kann nur die Benutzereinträge seines Aliaskataloges abspeichern. Beim Laden von Aliaskatalogeinträgen aus einer benutzereigenen Aliaskatalogdatei werden grundsätzlich wieder Benutzereinträge erzeugt.

Benutzereinträge

Benutzereinträge können vom Benutzer selbst erzeugt werden.

Die Benutzereinträge unterliegen den Konventionen zur Vereinbarung von Aliasnamen und werden entsprechend der Einstellung der taskbezogenen ACS-Optionen überprüft. Sie realisieren im Wesentlichen individuelle Dateinamensersetzungen.

Systemeinträge

Systemeinträge können nur von einem ACS-Administrator selbst erzeugt werden. Der Benutzer kann sie nur durch Laden aus einer privilegierten AC-Systemdatei erzeugen. Die Systemeinträge sind in einer AC-Systemdatei hinterlegt. Sie ist eine vom ACS-Administrator angelegte Aliaskatalogdatei, aus der der Benutzer Einträge in seinen Aliaskatalog laden kann. Die AC-Systemdatei muss vom Benutzer aus Gründen der Datensicherheit über eine vom ACS-Administrator vereinbarte Identifikation angesprochen werden.

Die Systemeinträge beschreiben die vom ACS-Administrator vordefinierte Softwarekonfiguration des Systems (Übersetzer, Binder, Dienstprogramme, ...).
Systemeinträge können vom Benutzer nicht geändert und nicht in eine Aliaskatalogdatei gespeichert werden.

Hinweise zur Versionsabhängigkeit

- AC-Dateien, die in Vorgängerversionen erstellt wurden, können in der aktuellen BS2000-Version problemlos aufgerufen werden.
- AC-Dateien, die in in der aktuellen BS2000-Version erstellt wurden, können auch in Vorgängerversionen aufgerufen werden.

Bildung von Aliasnamen

- Ein Aliasname ist ein beliebiger Name einer Datei/JV, den der Benutzer an Stelle des realen Namens verwenden kann. Aliasnamen haben dieselbe Syntax wie gewöhnliche Namen von Dateien/JVs (siehe „Regeln für die Bildung von Dateinamen“ (Vollqualifizierter Dateiname)).
- Ein Aliasname kann, wenn der ACS-Administrator es zulässt, eine Katalog- und/oder eine Benutzerkennung enthalten.
- Ein Aliasname darf nicht teilqualifiziert und nicht mit Wildcards vereinbart werden.
- In einem Aliasnamen ist eine Generations-/Versionsangabe unzulässig, für den zugeordneten realen Namen gilt das Gleiche.
- Die Vereinbarung von Aliasnamen, die die Benutzerkennungen TSOS oder SYS* enthalten, ist nicht erlaubt. „$.“-Namen hingegen sind erlaubt ($ oder $. sind grundsätzlich verschieden von $TSOS, auch wenn DEFLUID=$TSOS ist, d.h. $EDT oder $.EDT sind als Aliasnamen zulässig!).
  Die Namen $MYUSERID.DATEI.123 und DATEI.123 werden auch dann als verschieden angesehen, wenn MYUSERID die Benutzerkennung der laufenden Task ist.
- Angaben der DEFLUID-Benutzerkennung (Dateinamen mit führendem $-Zeichen, die keinen Punkt enthalten), werden als gleichwertig zu den mit „$.“ beginnenden Namen behandelt.

- Die Namen temporärer Dateien/JVs („#“-Namen) können nicht als Aliasnamen vereinbart werden und für temporäre Dateien/JVs kann kein Aliasname vereinbart werden. Das Kennzeichen für temporäre Dateien/JVs (Anfangszeichen „#“) darf nicht für Aliasnamen verwendet werden.
- Ist in einem Kommando oder Makro ein Name mit Generationsnummer angegeben, wird die Generationsangabe vor der Suche im Aliaskatalog abgetrennt. Wird der so gewählte Name gefunden, wird die Generationsangabe dem zugeordneten realen Namen angefügt.
  Falls der Name zu lang wird (nach Komplettierung > 54 Zeichen), gilt er als unzulässig und wird abgewiesen.
- Namen von temporären Systemdateien („S.“-Dateien) werden nicht umgesetzt.

> ¡ Die Wirksamkeit des **Dateischutzes** im BS2000 wird durch ACS in keiner Weise eingeschränkt, da die Schutzattribute einer Datei (Kennwörter, BACL, GUARDS, ...) von BS2000 erst überprüft werden, nachdem der TSOSCAT-Eintrag über einen vollständigen Pfadnamen gelesen worden ist. Der ACS-Aufruf aber hat bereits vor diesem Zeitpunkt stattgefunden und zwischen den Berechtigungsprüfungen und der eigentlichen Verarbeitung der Datei (OPEN- oder CATAL-Zugriff) erfolgt keine weitere Aliasnamensersetzung.

Einschränkungen beim Einsatz von Aliasnamen

- Aliasnamen werden nur für eine Task vereinbart und sind nur während der Laufzeit dieser Task gültig. Sie werden nicht im Katalogeintrag der Datei/JV hinterlegt, d.h. eine Datei/JV kann über beliebig viele unterschiedliche Aliasnamen angesprochen werden, bzw. verschiedene Tasks können mit dem gleichen Aliasnamen unterschiedliche Dateien/JVs bezeichnen. Für Dateinamen, die über Taskgrenzen weitergereicht werden, sollten daher keine Aliasnamen verwendet werden.
- Um Inkonsistenzen zu vermeiden, dürfen für Taskfamilien, die mit gleichen Programmbibliotheken arbeiten müssen, Aliasnamen nur über LOGON-Prozeduren vereinbart werden.
- Bei allen Aufrufen einer Datei/JV muss immer derselbe Name verwendet werden. Wird nach dem Kommando SHOW-FILE-ATTRIBUTES ein ersetzter und komplettierter Dateiname verwendet, kann es zu Inkonsistenzen kommen. Der komplettierte Dateiname kann eventuell als Aliasname für eine andere Datei/JV im Aliaskatalog gefunden und damit eine andere Datei/JV adressiert werden.
  Versehentliche Doppelersetzungen können verhindert werden, indem der Benutzer oder ACS-Administrator nur Dateinamen ohne Katalogkennung als Aliasnamen definiert.
- Ist für eine Task ein Aliaskatalog angelegt oder ist Präfixergänzung vereinbart, muss bei eventuellen Vergleichen von angegebenen Namen mit den vom Kommando SHOW-FILE-LINK bzw. Makro RDTFT oder vom OPEN im TU-FCB bereitgestellten Namen bei Dateien/JVs, für die Aliasadressierung nicht ausdrücklich untersagt ist, mit Ungleichheit gerechnet werden.
- Aliasnamen sollten, soweit irgend möglich, nicht mit Benutzerkennung – oder komplett mit Katalog- und Benutzerkennung – festgelegt werden.
  Der ACS-Administrator kann die Verwendung von Katalog- und Benutzerkennung in Aliasnamen für alle Benutzer verbieten (nicht gezielt für einzelne Benutzer steuerbar!).Außerdem ist bei Verwendung kompletter Pfadnamen als Aliasnamen die Gefahr von Doppelersetzungen deutlich erhöht.
- Der ACS-Administrator kann die Verwendung benutzereigener Aliaskataloginformationen auch ganz unterbinden. Entsprechende Kommandos werden dann nur ausgeführt, wenn die Task das ACS-Administrator-Privileg besitzt. Dies betrifft die Kommandos ADD-ALIAS-CATALOG-ENTRY, MODIFY-ALIAS-CATALOG-ENTRY und LOAD-ALIAS-CATALOG FROM-FILE=...

- DELETE-FILE oder DELETE-JV gibt vor dem Löschen von Dateien/JVs, die über Aliasnamen adressiert oder durch Präfix-Ergänzung verändert wurden, generell den realen Namen in einer beantwortbaren Meldung aus, wenn nicht CHECK=NO angegeben ist.
- REPAIR-DISK-FILE: Bei der Erzeugung der Fehlerdatei S.<filename1>.REPAIR erfolgt keine Präfix-Ergänzung und keine ACS-Ersetzung.
- Katalogkennungen, für die eine RFA-Verbindung besteht, dürfen weder in Aliaskatalogeinträgen noch im Präfix verwendet werden. Das Einfügen eines Präfixes in einen für einen RFA-Auftrag bestimmten Dateinamen ist generell untersagt.
- Für SERSLOG-Dateien dürfen keine Aliasnamen verwendet werden, die der Struktur der SERSLOG-Dateinamen entsprechen.
- Bei ELFE, SERSLOG usw. wird – falls die dem angegebenen Aliasnamen zugeordnete Datei nicht verfügbar ist – in der entsprechenden Fehlermeldung der Aliasname ausgegeben.
- **Nicht** über Aliasnamen adressiert werden dürfen:
  - Aliaskatalogdateien (ACFs und ACSFs) bei den Kommandos STORE-ALIAS-CATALOG und LOAD-ALIAS-CATALOG
  - USER-RESOURCES-FILEs im Kommando PRINT-DOCUMENT
  - alle DSSM-Dateien (Subsystembibliotheken, Rep- und Syntaxdateien, DSSM-Kataloge, usw.)
  - UDS-Datenbankdateien (Katalog, Realm, Afim-Datei, SLF-Datei, Status-Datei, Backout-Datei, Tempo-Datei, Hash-Bibliothek)
  - Dateien, die von TCP-IP bearbeitet werden
  - SATUT-Dateien für ausgewählte Informationen
  - SESAM-Datenbankdateien (ZD-Datei, ORG-Datei, Katalog-Datei, Logging- und Cursor-Dateien)
  - Dateien, die mit LEASY oder über File-Transfer bearbeitet werden sollen
  - bei der Systemeinleitung (Startup) verwendete Dateien

5.5.4 Präfix-Einfügung

Mit ACS ist es möglich, für alle Namen von Dateien/JVs, die durch den Aliaskatalog nicht erfasst werden, ein Präfix zu vereinbaren, das dem Namen vorangestellt wird.

Dadurch kann der Benutzer einen „Unterkatalog“ zu seiner Benutzerkennung bilden, indem er Dateien/JVs erzeugt, bearbeitet und löscht, die von anderen Dateien/JVs seiner Benutzerkennung, die nicht zu diesem Unterkatalog gehören, abgegrenzt sind.

Ein solcher Unterkatalog wird durch einen teilqualifizierten Namen (das letzte Zeichen muss ein Punkt sein) identifiziert. Dieser teilqualifizierte Name wird bei allen Kommandos und Makros den eingegebenen Namen von Dateien oder JVs vorangestellt, sofern diese keine Benutzerkennung enthalten oder nicht durch Aliaskatalogeinträge bereits erfasst sind.

Eine Einsatzmöglichkeit dieser Präfixfunktion ist beispielswiese dann gegeben, wenn sich mehrere Benutzer eine BS2000-Benutzerkennung teilen. Durch eine entsprechende Präfix-Regelung (z.B. Präfix=Name/Projekt) können diese Benutzer jeweils in einem eigenen Subkatalog arbeiten und Kommandos und Programme zur Dateiverarbeitung anwenden, ohne das Präfix selbst jedem Namen voranstellen zu müssen. Es können auch dieselben Prozeduren mit „festen“ Namen ohne Änderung auf jeden beliebigen Unterkatalog angewendet werden.

Die Regeln für die Präfix-Einfügung sind folgendermaßen definiert:

Ist der Name der Datei/JV im Aliaskatalog der Task als Aliasname eingetragen, wird kein Präfix vorangestellt, d.h diese Dateien/JVs können auch weiterhin ohne Benutzerkennung angesprochen werden (z.B. `START-EXECUTABLE-PROGRAM FROM-FILE=EDT`), ohne dass hierbei ein Präfix eingefügt wird.

Wenn der Datei-/JV-Name eine Benutzerkennung ungleich derjenigen der laufenden Task enthält, wird kein Präfix eingefügt.

Wenn der Datei-/JV-Name die eigene Benutzerkennung enthält, wird das Präfix nur eingefügt, wenn es lediglich aus einer Katalogkennung besteht und der Dateiname keine Katalogkennung enthält.

Eine Katalogkennung kann allein (in der Form „:catid:“) als Präfix vereinbart werden. Es wird dann in der laufenden Task den Namen der Datei/JV, die sich auf die eigene Benutzerkennung beziehen, vorangestellt. Die Wirkung entspricht quasi einer temporären, tasklokalen Änderung der Default-Katalogkennung. Es dürfen grundsätzlich nur systemlokale Katalogkennungen (Pubsets des eigenen Systems) angegeben werden; RFA wird von ACS nicht unterstützt.

Wenn der Name und das Präfix eine Katalogkennung enthalten, findet keine Einfügung statt.

Namen mit der Benutzerkennung TSOS sind generell von der Präfix-Ergänzung ausgeschlossen.

Enthält der Name der Datei/JV eine Katalogkennung (aber keine Benutzerkennung), wird das Präfix unmittelbar hinter der Katalogkennung eingefügt.

Wenn nach der Präfix-Einfügung der Name der Datei/JV zu lang wird (> 54 Zeichen), gilt er als fehlerhaft und wird zurückgewiesen.

Durch eine Option kann gesteuert werden, ob das Präfix auch dann verwendet werden soll, wenn der Name der Datei/JV bereits mit der als Präfix vereinbarten Zeichenfolge beginnt.

Einschränkungen beim Voranstellen eines Präfixes

Ist für eine Task die Namensergänzung durch Präfix vereinbart, wird das Präfix allen Namen vorangestellt, die ohne Benutzerkennung angegeben werden und nicht im Aliaskatalog vermerkt sind.

> ! Achtung bei Systemdateien!
> Aus `START-EXECUTABLE-PROGRAM FROM-FILE=EDT` wird: `START-EXECUTABLE-PROGRAM FROM-FILE=<Präfix.>EDT`

Soll dem Namen einer Datei/JV ein Präfix vorangestellt werden, darf der Name nicht als Aliasname im Aliaskatalog der Task enthalten sein.

Das dem Namen vorangestellte Präfix ist Bestandteil des Namens, unter dem die Datei /JV im Dateikatalog TSOSCAT katalogisiert ist. Das bedeutet, dass bei allen weiteren Zugriffen auf diese Datei/JV dieses Präfix mit ACS vereinbart oder explizit als Bestandteil des Namens angegeben werden muss.

Der angegebene Name muss auch dann den Namenskonventionen entsprechen, wenn später noch ein Präfix vorangestellt wird.

Wird bei der Bearbeitung von DAMP-Dateien, die unter der eigenen Benutzerkennung angelegt sind, mit Präfixvereinbarung gearbeitet, werden auch alle DAMP-internen Dateien mit vorgeschaltetem Präfix adressiert.

5.6 Dateisperren-Verwaltung

Das Kommando SHOW-FILE-LOCKS zeigt für eine Datei die aktuell wirksamen Sperren an. Dateisperren können durch folgende Zustände/Operationen verursacht werden:

- Die Datei ist gerade geöffnet.
- Für die Datei wurde eine explizite Reservierung vorgenommen (Kommando SECURE-RESOURCE-ALLOCATION).
- Für die Datei wurde zum Ausdrucken eine Sperre vereinbart (mit dem Kommando PRINT-DOCUMENT ...,LOCK-FILE=*YES, dem Makro PRNT oder mit dem Kommando MODIFY-PRINT-JOB-ATTRIBUTES).
- Die Datei ist für eine Dateiübertragung mit FT (File-Transfer) reserviert (Kommando TRANSFER-FILE).
- Eine SYSLST-Datei wartet nach einem EXIT-JOB-Kommando auf das Ausdrucken.
- Die Datei wird gerade von einem CCOPY-Auftrag bearbeitet.
- Die Datei ist für einen Zugriff durch EAM reserviert.
- Ein kurzfristiger Verbindungsausfall in einem Rechnerverbund oder ein Systemfehler am lokalen System führen zu „unberechtigten Dateisperren“ (siehe unten).

Das Kommando SHOW-FILE-LOCKS zeigt nicht die Sperren der Katalogverwaltung an.

SHOW-FILE-LOCKS dient dem Eigentümer einer Datei und dem Systembetreuer dazu, die Ursache eines Verarbeitungsproblems wegen einer bestehenden Dateisperre zu diagnostizieren. Berechtigt zur Eingabe des Kommandos ist der Eigentümer der Datei, andere Benutzer, die eine Zugriffsberechtigung auf diese Datei haben, und Kennungen mit dem Privileg TSOS.

Rücksetzen unberechtigter Dateisperren

Sog. unberechtigt sitzende Dateisperren können das (Weiter-)Arbeiten mit einer Datei verhindern. Ursachen für solche Dateisperren sind z.B. ein kurzfristiger Verbindungsausfall in einem Rechnerverbund oder ein Systemfehler am lokalen System, der das Rücksetzen einer Dateisperre verhindert.

Das Kommando REPAIR-FILE-LOCKS erlaubt die Zurücksetzung von unberechtigt sitzenden Dateisperren für den Eigentümer der Datei und die Systembetreuung.

5.7 PFA: Performant File Access

Im Rahmen des Hiperfile-Konzepts wurden die performance-bezogenen Dateiattribute PERFORMANCE, USAGE und DISK-WRITE eingeführt:

- PERFORMANCE
  spezifiziert die gewünschte Performance-Güte bzgl. des Ein-/Ausgabeverhaltens bei Dateizugriffen
- USAGE
  beschreibt, ob die Performance nur für lesende, nur für schreibende oder beide Arten von Zugriffen gewünscht wird
- DISK-WRITE
  bestimmt, zu welchem Zeitpunkt der Dateibearbeitung mit erhöhter Performance-Güte Datenkonsistenz für Schreiboperationen gegeben sein muss

PFA bei SF-Pubsets

Die performance-bezogenen Dateiattribute steuern bei SF-Pubsets die Nutzung der Caches: Eine Datei, für die eine erhöhte Performance-Anforderung besteht, wird automatisch mit Cache-Pufferung bearbeitet. Dabei legen die USAGE- und DISK-WRITE-Vorgaben und der dem Pubset zugeordnete Cache-Typ (VOLATILITY) fest, ob ein Lese- und/oder ein Schreib-Caching erfolgt.

Werden die Dateiattribute PERFORMANCE=*HIGH, USAGE=*READ-WRITE und DISK-WRITE=*IMMEDIATE für eine Datei gewählt, die auf einen SF-Pubset mit einem nicht-schreibsicheren Cache (z.B. Hauptspeicher-Cache) gelegt wird, so erfolgt die Cache-Zwischenpufferung nur lesend, da die Voraussetzungen für ein ausfallsicheres Caching der Schreib-Ein-/Ausgaben nicht erfüllt sind. Es obliegt dem Benutzer selbst, sicherzustellen, dass die von ihm vorgegebenen Dateiattribute mit den Cache-Eigenschaften harmonieren.

Das Kommando SHOW-MASTER-CATALOG-ENTRY INF=*USER zeigt über die Ausgabezeile CACHE-MEDIUM an, ob ein nicht-flüchtiges Cache-Medium dem Pubset zugeordnet ist. Bei der Ausgabe NONVOLATILE ist dies der Fall; nur dann kommen die Dateiattribute USAGE=*WRITE und DISK-WRITE=*IMMEDIATE zur Wirkung.

> ¡ Ab BS2000 OSD/BC V11.0 entfällt die Unterstützung von nicht-flüchtigen Cache-Medien.

Für SF-Pubsets, die keinen Cache zugeordnet haben, sind die Attribute PERFORMANCE, USAGE und DISK-WRITE ohne Relevanz. Das Performance-Verhalten ist bereits vollständig durch den Ablageort bestimmt.

Diese Beispiele zeigen, dass der Benutzer die Merkmale der verschiedenen Medien sehr genau kennen muss, um das von ihm gewünschte Performance-Verhalten tatsächlich zu erreichen.

PFA bei SM-Pubsets

Die performance-bezogenen Dateiattribute dienen bei SM-Pubsets der Volume-Set-Selektion zur Bestimmung des zur Dateiablage am besten geeigneten Volume-Sets: Bei der Neuanlage einer Datei wird diese automatisch anhand ihrer Performance-Attribute auf ein Volume-Set gebracht, dessen Performance-Profil den Dateianforderungen am besten genügt.

Das Performance-Profil eines Volume-Sets kann durch das Kommando SHOW-PUBSET-DEFINITION-FILE abgefragt werden. Es ergibt sich aus dem Performance-Spektrum eines Volume-Sets und der Einschränkung bzgl. der Schreibsicherheit bei erhöhten Performan-ce-Werten. Die Komplexität ist notwendig, da das Performance-Profil die unterschiedlichen Cache- und Plattentypen zu charakterisieren hat. Der Ausprägung der performance-relevanten Attribute eines Volume-Sets liegen folgende Überlegungen zu Grunde:

Ein z.B. mit einem Cache ausgestatteter Volume-Set bietet bei Dateibearbeitung ohne Cache normale Performance (STD), bei Dateibearbeitung mit Cache erhöhte Performance (HIGH bzw. VERY-HIGH). Ein Volume-Set mit Cache verfügt also über ein Performance-Spektrum (von STD über HIGH bis zu VERY-HIGH).

> ¡ Ab BS2000 OSD/BC V11.0 entfällt die Unterstützung von nicht-flüchtigen Cache-Medien.

Das Kommando SHOW-PUBSET-DEFINITION-FILE liefert bei Volume-Sets eine Liste von Performance-Werten. Allerdings kann bei nicht-schreibsicheren Caches die erhöhte Performance nur bei Dateibearbeitungen geboten werden, bei denen keine Datenkonsistenz nach jeder Schreiboperation gefordert wird.
Diese Einschränkung bzgl. erhöhter Performance wird durch das performance-relevante Attribut WRITE-CONSISTENCY ausgedrückt:
WRITE-CONSISTENCY=*BY-CLOSE bringt zum Ausdruck, dass erhöhte Performance nur unter Verzicht auf Schreibsicherheit möglich ist.

Es obliegt dem Systembetreuer, die dem Performance-Profil eines Volume-Sets adäquaten Hardware- bzw. Cache-Konfigurationen durch das Kommando MODIFY-PUBSET-CACHE-ATTRIBUTES bereitzustellen.

5.7.1 Das HIPERFILE-/PFA-Konzept

Unter dem Begriff „HIPERFILE-Konzept“ (High Performance Files) werden im BS2000 verschiedene Erweiterungen im Bereich der Systemsoftware und der Hardware zusammengefasst, die durch „Caching“ von Dateien den Dateizugriff beschleunigen und einen eventuellen I/O-Engpass vermeiden sollen. Als Puffer- (bzw. Cache-)Speicher kommen Halbleiter-Speicher mit kurzen Zugriffszeiten zum Einsatz, die eine Anpassung zwischen der Zugriffszeit zum Hauptspeicher und der (längeren) Zugriffszeit zu Festplattenspeichern ermöglichen.

Das HIPERFILE-Konzept des BS2000 bietet Caching sowohl über die Kommando-Oberfläche des jeweils beteiligten Subsystems als auch über eine in das DVS integrierte einheitliche Kommando-Oberfläche:

- ADM-PFA (Administrator Performant File Access):
  Caching in den Medium Hauptspeicher über privilegierte Kommandos des DAB
- PFA (User Performant File Access):
  Caching über die Einbettung der Hiperfiles in das DVS

Mit ADM-PFA-Caching wird die Verwendung der Konzepte, Methoden und Kommandos des Cache-Handler-Subsystems DAB (START-DAB-CACHING) bezeichnet, um sie von der Einbettung der Hiperfiles in das DVS, das sog. PFA-Konzept, begrifflich abzugrenzen. ADM-PFA-Caching wird im Handbuch zu „DAB“ [5] beschrieben.

BS2000 unterstützt im Rahmen des PFA-Konzeptes das Cache-Medium Hauptspeicher. Als Treibersoftware zur Bedienung dieses Mediums wird das Subsystem DAB als Cache-Handler benötigt:

Die Einbettung der Hiperfiles in das DVS wird zum einen dadurch erreicht, dass Pubsets Cache-Medien zugeordnet werden können und zum anderen dadurch, dass der Benutzer seine Dateien durch Vergabe von performance-bezogenen Attributen zu Hiperfiles erklären kann.

5.7.2 Die Auswahl der Pubsets

Mit dem Kommando SHOW-MASTER-CATALOG-ENTRY (Operand INFORMATION= *USER) kann sich der Benutzer über die aktuell gültige und aktivierte Cache-Zuordnung (kein Cache oder Cache) bei importierten SF-Pubsets informieren.

Über das Performance-Profil von Volume-Sets bei SM-Pubsets gibt das Kommando SHOW-PUBSET-DEFINITION-FILE Auskunft. Darüber hinaus kann der Benutzer mit dem Kommando SHOW-PUBSET-FILE-SERVICES verifizieren, ob der Pubset den – seinen gewählten Dateieigenschaften entsprechenden – Service bieten kann.

Eine Voraussetzung zur Nutzung von Hiperfiles ist es, dass dem Benutzer die dazu erforderliche Berechtigung (DMS-TUNING-RESOURCES) vom Systembetreuer erteilt wurde und zusätzlich bei SM-Pubsets dem jeweiligen Benutzerkontingent (HIGH-PERF-SPACE oder VERY-HIGH-PERF-SPACE) ein Wert größer null zugewiesen wurde.
Auskunft darüber gibt das Kommando SHOW-USER-ATTRIBUTES.

Die Systembetreuung kann dem einzelnen Benutzer folgende Caching-Berechtigungen verleihen:

- NONE: der Benutzer erhält keine Berechtigung, Dateien über Caches zu bearbeiten.
- CONCURRENT-USE: der Benutzer ist berechtigt, Dateien über einen Cache zu bearbeiten, steht dabei aber in Konkurrenz zu allen anderen Benutzern dieses Pubsets mit der gleichen Berechtigung. Bei knappem Speicherplatz im Cache-Bereich können also Teile der Benutzerdatei aus dem Cache verdrängt werden.
- EXCLUSIVE-USE: der Benutzer ist berechtigt, Dateien über einen Cache bevorzugt zu bearbeiten. Auch bei knappem Speicherplatz versucht das System genau dann die referenzierten Daten der Datei des Benutzers immer vollständig im Cache-Bereich des Pubsets zu halten, wenn der Benutzer dies durch ein entsprechendes Datei-Attribut anfordert.

Im Anschluss an den folgenden Abschnitt „Die Auswahl der Dateien“ werden in Form einer Tabelle die Auswirkungen dieser Berechtigungen in Zusammenspiel mit den Datei-Attributen des Benutzers erläutert.

5.7.3 Die Auswahl der Dateien

Hat die Systembetreuung einem Benutzer die Berechtigung eingeräumt, seine Dateien über ein Cache-Medium verarbeiten zu lassen, kann der Benutzer seine Ein-/Ausgabe-kritischen Dateien durch entsprechende Dateiattribute kennzeichnen. Die Auswahl der Dateien sollte gezielt unter dem Gesichtspunkt der Reduzierung von Ein-/Ausgaben erfolgen,
d.h. solche Dateien müssen ausgewählt werden, bei denen ein unverhältnismäßig hoher Anteil von DVS-Ein-/Ausgaben zu beobachten ist (als Entscheidungshilfe kann hier das Messsystem SM2 eingesetzt werden).

Mit den Kommandos CREATE-FILE, MODIFY-FILE-ATTRIBUTES und ADD-FILE-LINK bzw. den Makros FILE und CATAL kann der Benutzer für seine ausgewählten Dateien die Attribute PERFORMANCE, USAGE und DISK-WRITE vergeben.

Die Dateieigenschaft PERFORMANCE beschreibt die Wertigkeit der angestrebten Ein-/Ausgabe-Performance-Verbesserung und kann mit folgenden Werten belegt werden:

- STD
  Die Datei wird ohne Nutzung eines Cache bearbeitet.
- HIGH
  Die Datei soll über einen Cache bearbeitet werden, kann aber bei knappem Speicher teilweise verdrängt werden. Diese Angabe wird nur dann wirksam, wenn für die Benutzerkennung im Benutzerkatalog des Pubsets die Berechtigungen CONCURRENT-USE oder EXCLUSIVE-USE eingetragen sind.
- VERY-HIGH
  Die Datei soll auch bei knappem Speicher permanent im Cache gehalten werden. Diese Angabe wird nur dann wirksam, wenn für die Benutzerkennung im Benutzerkatalog des Pubsets die Berechtigung EXCLUSIVE-USE eingetragen ist.
- USER-MAXIMUM
  Die Datei soll das höchste Ein-/Ausgabe-Attribut erhalten, das für den Benutzer im Benutzerkatalog eingetragen ist.

Die Dateieigenschaft USAGE bezeichnet den Cache-Modus und kann mit folgenden Werten angegeben werden:

- READ
  Die erhöhte Performance-Anforderung soll nur für Leseoperationen gelten (Lese-Cache).
- WRITE
  Die erhöhte Performance-Anforderung soll nur für Schreiboperationen gelten (Schreib-Cache).
- READ-WRITE
  Die Anforderungen gelten für Lese- und Schreiboperationen (Schreib-Lese-Cache).

Die Dateieigenschaft DISK-WRITE gibt an, zu welchem Zeitpunkt nach einer Schreiboperation Datenkonsistenz gefordert wird. Implizit kann der Benutzer über diesen Operanden den Wunsch formulieren, seine Dateien in einem sicheren, nicht-flüchtigen Cache-Medium zu verarbeiten.

- IMMEDIATE
  Die Daten der Datei müssen sich direkt nach Beendigung einer Schreiboperation in einem konsistenten Zustand befinden, d.h. implizit, dass die Datei nicht über einen flüchtigen Schreib-Cache bearbeitet werden soll. Das Zurückschreiben soll auf ein sicheres, nicht-flüchtiges Medium erfolgen.
  Für Dateien, die wichtige Daten enthalten, sollte deshalb Datenkonsistenz nach jeder Schreiboperation gefordert werden.

  > ¡ Ab BS2000 OSD/BC V11.0 entfällt die Unterstützung von nicht-flüchtigen Cache-Medien.

- BY-CLOSE
  Die Daten der Datei müssen sich erst nach der CLOSE-Verarbeitung in einem konsistenten Zustand befinden; der Benutzer verzichtet auf die Verarbeitung in einem nichtflüchtigen Speichermedium.
  Dabei ist zu beachten, dass bei der Bearbeitung über einen flüchtigen Schreib-Cache sich die Daten der Datei erst nach der CLOSE-Verarbeitung in einem konsistenten Zustand befinden. Systemfehler während der Bearbeitungsphase können zu Inkonsistenzen führen.
  Es sollten deshalb nur solche Dateien dieses Attribut erhalten, die ausschließlich während einer laufenden Anwendung relevant sind (z.B. Ergebnislisten einer Compilierung).

Der Benutzer kann die gewünschten Dateiattribute statisch und dynamisch vergeben. Die statische Vergabe der Dateiattribute geschieht über die Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES bzw. über die Makros CATAL und FILE. Für die Erzeugung des entsprechenden Katalogeintrages ist die Berechtigung des Dateieigentümers im Benutzerkatalog erforderlich. Fehlt die angeforderte Berechtigung, erhält die Datei die für die Benutzerkennung maximal zulässige Performance-Berechtigung.
Der Benutzer hat weiter die Möglichkeit, für eine spezielle Dateiverarbeitung die im Katalogeintrag hinterlegten Attribute außer Kraft zu setzen, ohne den Katalogeintrag selbst zu verändern (dynamische Vergabe der Dateiattribute). Mit dem Kommando ADD-FILE-LINK bzw. über die FCB-Schnittstelle können die Dateiattribute PERFORMANCE und USAGE jeweils in Richtung geringerer Wertigkeit verändert werden. Die im Dateikatalog hinterlegten Dateiattribute bleiben dabei unverändert.

Die aktuellen Performance-Eigenschaften werden beim Eröffnen der Datei vom DVS ausgewertet, wobei ein Abgleich der Angaben im Katalogeintrag, der dynamisch vereinbarten Werte und der Berechtigung im Benutzerkatalog erfolgt. Zugleich wird geprüft, ob die Angaben des Benutzers mit der gültigen Cache-Zuordnung des betreffenden Pubsets verträglich sind. Bei Verträglichkeit erfolgt die Verarbeitung der Datei im Cache. Anderenfalls wird die Datei ohne Cache verarbeitet; eine Zurückweisung des OPEN auf Grund mangelnder Berechtigung oder unverträglicher Konfiguration erfolgt nicht.

Die folgende Tabelle zeigt, wie die Angaben der Systembetreuung für die Benutzerkennung im Benutzerkatalog und die Angaben des Benutzers für seine Dateien zusammenwirken.

Cached-Files	Berechtigung im Daten-Pubset	Erlaubte Performance-Maßnahmen für den Benutzer auf dem Daten-Pubset
BY-USER-SELECTED	NONE	keine
	CONCURRENT-USE	Dateiattribut PERFORMANCE=HIGH
	EXCLUSIVE-USE	Dateiattribut PERFORMANCE=HIGH/VERY-HIGH
ALL / BY-SYSTEM	NONE	Dateiattribut PERFORMANCE=HIGH

Mit dem Kommando START-FILE-CACHING kann nachträglich für bereits geöffnete Dateien Caching gestartet werden. Über die Operanden PERFORMANCE und USAGE werden die Dateieigenschaften Performance-Wertigkeit und Cache-Modus bestimmt. Voraussetzungen für das nachträgliche Zuschalten von Caching sind:

- Der Aufrufer besitzt die Zugriffsberechtigung auf die Datei besitzt (Eigentümer der Datei oder die Systembetreuung)
- Für den Pubset, auf dem sich die Datei befindet, gilt:
  - Der Pubset besitzt eine gültige und aktivierte Cache-Zuordnung.
  - Der Pubset muss lokal zugreifbar sein.
- Das Kommando muss an demselben System eingegeben werden, wenn der dem Pubset zugeordnete Cache systemlokal betrieben wird.
- Dem Aufrufer wurde die erforderliche Cache-Berechtigung von der Systembetreuung erteilt.

Ist die Datei nicht geöffnet oder wird sie bereits mit Cache bearbeitet, wird das Kommando START-FILE-CACHING abgewiesen.

Das Kommando STOP-FILE-CACHING beendet das Caching für eine geöffnete Datei oder eine Datei, von der sich noch Daten im Cache befinden. Die noch im Cache befindlichen Daten werden zurückgeschrieben (Ausnahme: reiner Lese-Cache) und der Cache-Inhalt invalidiert. Zum Ausführen des Kommandos gelten die gleichen Voraussetzungen wie bei START-FILE-CACHING.

5.7.4 Fehler beim Schließen eines Hiperfiles

Mit „Hiperfile“ wird im Folgenden eine Datei bezeichnet, die auf einem PFA-Pubset liegt und das gültige Performance-Attribut HIGH oder VERY-HIGH besitzt, oder die auf einem PFA-Pubset liegt, der das Cache-Attribut CACHED-FILES=*ALL besitzt.

Beim Schließen (CLOSE-Verarbeitung) eines Hiperfiles werden alle von der Anwendung geschriebenen Blöcke der Datei, die noch nicht von einem Cache-Handler bzw. einer Cache-Steuerung vom Cache auf die entsprechenden Platten des Pubsets gesichert worden sind, explizit auf Platte geschrieben.

Tritt bei dieser Sicherungsaktion ein Fehler auf, weil ein angesprochenes Plattengerät einen Defekt hat, so meldet die CLOSE-Verarbeitung den Fehlercode

```
DMS0E27:  Fehler beim Schliessen einer Datei. Ein-/Ausgabe mit Hardware-Fehler
          beendet.
```

Die Daten sind jedoch noch lesbar im Cache vorhanden, der Cache wird in der Folge nicht abgebaut, und die im Cache verbliebenen Daten können nach Behebung des Plattendefekts ohne Einwirkung von außen auf die Platte abfließen. Die Datei kann in der Regel wieder geöffnet und bearbeitet werden, obwohl Daten im Cache nicht zurückgeschrieben werden konnten. Solche Dateien können mit folgendem Kommando angezeigt werden:
SHOW-FILE-ATTRIBUTES ...,STATUS=*PAR(CACHE-NOT-SAVED=*YES).

Dateien, die wegen Nutzung eines Caches noch kein konsistentes Abbild auf Platte haben, kennzeichnet das Betriebssystem als „Datei mit Daten im Cache“; solche Dateien (Hiperfiles) kann man sich mit folgendem Kommando anzeigen lassen:
SHOW-FILE-ATTRIBUTES ..., STATUS=*PAR(CACHED=*YES)

Hierbei werden auch diejenigen Dateien mitaufgelistet, die aktuell gecachet werden (auch mit reinem Lese-Cache) und zurzeit geöffnet sind.

Die Tatsache, dass nicht erfolgreich auf Platte geschriebene Blöcke im Cache noch lesbar sind, kann man sich wie folgt zu Nutze machen:

Ein Wieder-Eröffnen der Datei ist jederzeit möglich. Mit einem COPY-FILE Kommando kann die Datei auf einen anderen Datenträger „gerettet“ werden; hierzu ist es notwendig, vor dem COPY-FILE Kommando die Zieldatei mit einem expliziten CREATE-FILE Kommando anzulegen.

Mit einem DELETE-FILE Kommando kann eine solche Datei logisch gelöscht werden, dabei findet gleichzeitig ein Invalidieren der im Cache verbliebenen Daten dieser Datei statt.

Ist auch nach einem Techniker-Eingriff das Sichern der im Cache verbliebenen Daten auf Platte nicht mehr möglich, so kann der Cache mit dem Kommando FORCE-DESTROY-CACHE ohne Sichern der Daten aufgelöst werden.

Diejenigen Hiperfiles des Pubsets, die als „Datei mit Daten im Cache“ gekennzeichnet sind, werden beim nächsten IMPORT-PUBSET im Katalog so gekennzeichnet, dass sie (wegen möglicher Dateninkonsistenz) nicht mehr geöffnet werden können; Voraussetzung hierfür ist, dass die Datei das Performance-Attribut DISK-WRITE=*IMMEDIATE besitzt.
Die Menge derjenigen Hiperfiles, die wegen möglicher Dateninkonsistenz nicht mehr geöffnet werden können, kann man sich mit folgendem Kommando auflisten lassen:
SHOW-FILE-ATTRIBUTES ...,STATUS=*PAR(OPEN-ALLOWED=*NO)

Solche Dateien sollten gelöscht werden. Ein REPAIR-DISK-FILES-Kommando (nur unter TSOS-Privileg möglich) setzt das Attribut OPEN-ALLOWED=NO zwar zurück und die Datei kann wieder eröffnet werden, eine Aussage über die Datenkonsistenz kann dabei aber nicht gemacht werden! Ist ein Plattendefekt nicht zu beheben, so muss gegebenenfalls das ganze Pubset neu initialisiert werden.

5.7.5 Beispiele

Die folgenden beiden Beispiele sollen den Ablauf aus Sicht des Benutzers, einmal für einen SF-Pubset und einmal für einen SM-Pubset, zusammenfassen.

Beispiel 1: SF-Pubset

Der importierte Daten-Pubset DAT1 (SF-Pubset) ist im Hauptspeicher gepuffert.

```
/SHOW-MASTER-CATALOG-ENTRY CATALOG-ID=DAT1, INFORMATION=*USER ————————  (1)
PUBSET DAT1: SINGLE-FEATURE, PUBRES-UNIT=FEE8, LOCAL-IMPORTED, NK-FORMAT
             EXTRA-LARGE-CATALOG
---- CURRENT PUBSET-PARAMETERS      ---------------------------------------
 MAXIMAL I/O LENGTH                   | 240 HP
 ALLOCATION UNIT SIZE                 | 3   HP
 PHYSICAL ALLOCATION                  | BY ADMINISTRATOR
 SPEEDCAT MODE                        | SCA RUNNING
---- CURRENT CACHE-CONFIGURATION    ---------------------------------------
 CACHE MEDIUM                         | VOLATILE
 CACHE SIZE                           | 80 MB
 DOUBLE BUFFERING                     | NO
/SHOW-USER-ATTRIBUTES USER2, PUBSET=DAT1,INFORMATION=*PUBSET-ATTRIBUTES ——(2)
SHOW-USER-ATTRIBUTES --- PUBSET DAT1 - USER USER2         2017-03-20 12:56:15
------------------------------------------------------------------------------
USER-ID                   USER2           PUBLIC-SPACE-EXCESS     *NOT-ALLOWED
CODED-CHARACTER-SET     EDF03IRV          DMS-TUNING-RESOURCES    *CONCURRENT
NET-CODED-CHAR-SET      *ISO              PHYSICAL-ALLOCATION     *NOT-ALLOWED
                                        NET-STORAGE-USAGE       *ALLOWED
FILE-NUMBER-LIMIT         16777215        JV-NUMBER-LIMIT           16777215
LIMITED FILES                   27        JOB-VARIABLES                    0
PERM-SPACE-LIMIT          16777215        TEMP-SPACE-LIMIT        2147483647
PERM-SPACE-USED             424572        TEMP-SPACE-USED              12000
------------------------------------------------------------------------------
SHOW-USER-ATTRIBUTES           END OF DISPLAY FOR USER USER2  ON PUBSET DAT1
/SHOW-FILE-ATTRIBUTES FILE-NAME=:DAT1:USERFILE ————————————————————  (3)
0000012000 :DAT1:$USER2.USERFILE
  ------------------------------ ORGANIZATION ------------------------------
  FILE-STRUC = NONE        BUF-LEN    = NONE        BLK-CONTR  = NONE
  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
  REC-FORM   = NONE        REC-SIZE   = 0
  AVAIL      = *STD
  WORK-FILE = *NO          F-PREFORM = *NK2         S0-MIGR = *ALLOWED
:DAT1: PUBLIC:      1 FILE  RES=     12000 FRE=     12000 REL=    12000 PAGES
/MODIFY-FILE-ATTRIBUTES FILE-NAME=:DAT1:USERFILE,
     IO-ATTRIBUTES=*PARAMETERS(PERFORMANCE=*HIGH,
     USAGE=*READ),DISK-WRITE=*IMMEDIATE  ———————————————————————————— (4)
```

(1) Mit dem Kommando SHOW-MASTER-CATALOG-ENTRY kann sich der Benutzer über den für den Pubset DAT1 zugeschalteten Cache insoweit informieren, dass er angezeigt bekommt, ob ein Cache aktiv ist. In diesem Beispiel ist dies der Fall.

(2) Der Benutzer kann sich mit dem Kommando SHOW-USER-ATTRIBUTES über seine aktuell gültige, pubset-spezifische Berechtigung, Dateien über Caches verarbeiten zu lassen, informieren. Die angezeigte Berechtigung zeigt ihm die für ihn höchstens zulässige Berechtigung (CONCURRENT) an.

(3) Mit dem Kommando SHOW-FILE-ATTRIBUTES kann sich der Benutzer über die Dateiattribute informieren, die er für seine Datei USERFILE bislang vergeben hat.

(4) Der Benutzer selbst vergibt für seine Datei USERFILE die folgenden performance- relevanten Dateiattribute:

- Die Datei soll über einen Cache bearbeitet werden (PERFORMANCE=*HIGH). Diese Angabe wird wirksam, da für die Benutzerkennung im Benutzerkatalog die Berechtigung CONCURRENT eingetragen ist.
- Die Anforderung der hochperformanten Dateiverarbeitung soll nur für Leseoperationen gelten (USAGE=*READ).
- Wenn die Datei schreibend geöffnet wird, wird der Cache nicht genutzt, da mit DISK-WRITE=*IMMEDIATE eine stets konsistente schreibende Verarbeitung der Datei gefordert wird.

Beispiel 2: SM-Pubset

Der Volume-Set VS01 des importierten Daten-Pubsets SMS1 (SM-Pubset) ist im Hauptspeicher gepuffert.

```
/SHOW-PUBSET-DEFINITION-FILE PUBSET=SMS1,VOLUME-SET=VS01 ————————————  (1)
 ------------------------------------------------------------------------
 COMMAND: SHOW-PUBSET-DEFINITION-FILE
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 PUBSET SMS1: TYPE=SYSTEM-MANAGED, VOLUMESETS=3, DEFAULT FILE FORMAT=NK2
 ---- VOLUME-SET INFORMATION --------- + -----------------------------------
 VOLUME-SET VS01: NORMAL-USE, NK2-FORMAT
---- GLOBAL ATTRIBUTES      --------- + -----------------------------------
 AVAILABILITY                         | STANDARD
 USAGE                                | WORK
 FORMAT                               | NK2-FORMAT
 MAXIMAL I/O LENGTH                   | 240 HP
 ALLOCATION UNIT SIZE                 | 3   HP
 DRV-VOLSET                           | NO
 NEW FILE ALLOCATION                  | NOT RESTRICTED
 VOLUME SET ACCESS                    | NOT RESTRICTED
---- PERFORMANCE ATTRIBUTES --------- + -----------------------------------
 PERFORMANCE                          | STANDARD
 WRITE-CONSISTENCY                    | IMMEDIATE
```

(1) Mit dem Kommando SHOW-PUBSET-DEFINITION-FILE kann sich der Benutzer über das Performance-Profil des Volume-Sets VS01 des Pubsets SMS1 informieren.

```
/SHOW-USER-ATTRIBUTES USER2,PUBSET=SMS1,INFORMATION=*PUBSET-ATTRIBUTES ———(2)
SHOW-USER-ATTRIBUTES --- PUBSET SMS1 - USER  USER2        2017-03-21 11:53:55
------------------------------------------------------------------------------
USER-ID                 USER2         PUBLIC-SPACE-EXCESS     *NOT-ALLOWED
CODED-CHARACTER-SET     EDF03IRV        DMS-TUNING-RESOURCES    *CONCURRENT
NET-CODED-CHAR-SET      *ISO            PHYSICAL-ALLOCATION     *NOT-ALLOWED
DEF-STORAGE-CLASS       *NONE           NET-STORAGE-USAGE       *ALLOWED
FILE-NUMBER-LIMIT         16777215      JV-NUMBER-LIMIT           16777215
LIMITED FILES                    1      JOB-VARIABLES                    0
                                    PERM-SPACE      TEMP-SPACE      WORK-SPACE
TOTAL-SPACE             LIMIT      2147483647      2147483647      2147483647
                        USED                0               0               3
S0-LEVEL-SPACE          LIMIT        16777215
                        USED                0
HIGH-PERF-SPACE         LIMIT        *MAXIMUM        *MAXIMUM        *MAXIMUM
                        USED                0               0               0
VERY-HIGH-PERF-SPACE    LIMIT        *MAXIMUM        *MAXIMUM        *MAXIMUM
                        USED                0               0               0
HIGH-AVAILABLE-SPACE    LIMIT               0
                        USED                0
------------------------------------------------------------------------------
SHOW-USER-ATTRIBUTES          END OF DISPLAY FOR USER USER2    ON PUBSET SMS1
```

(2) Der Benutzer kann sich mit dem Kommando SHOW-USER-ATTRIBUTES über seine aktuell gültige, pubset-spezifische Berechtigung, Dateien über Caches verarbeiten zu lassen, informieren. Die angezeigte Berechtigung zeigt ihm die für ihn höchstens zulässige Berechtigung (CONCURRENT) an. Gleichzeitig wird ihm angezeigt, dass die Kontingente für HIGH-PERF-SPACE und VERY-HIGH-PERF-SPACE einen Wert größer null haben und er die ihm zugewiesenen Kontingente noch nicht überschritten hat.

```
/SHOW-PUBSET-FILE-SERVICES PUBSET=SMS1, SELECT=*BY-ATTRIBUTES(FILE-
ATTRIBUTES=*PARAMETERS( IO-
ATTRIBUTES=*PARAMETERS(PERFORMANCE=*HIGH,USAGE=*READ-WRITE), DISK-WRITE=*BY-
CLOSE))————————————————————————————————————————————————————————————————  (3)
 WORK-F  AVAIL F-FORM   IO(PERF)  IO(USAGE) DISK-WRITE   SUPPORT-QUALITY
------+------+------+----------+----------+----------+------------------
  YES    STD    NK2        HIGH READ-WRITE   BY-CLOSE           OPTIMAL
/CREATE-FILE FILE-NAME=:SMS1:USERFILE, IO-
ATTRIBUTES=*PARAMETERS(PERFORMANCE=*HIGH, USAGE=*READ-WRITE),DISK-WRITE=*BY-
CLOSE ————————————————————————————————————————————————————————————————————  (4)
```

(3) Mit dem Kommando SHOW-PUBSET-FILE-SERVICES kann der Benutzer verifizieren, inwieweit der Pubset SMS1 den von ihm geforderten Datei-Service bietet.

(4) Der Benutzer selbst richtet seine Datei USERFILE mit folgenden performance-relevanten Dateiattributen ein:

- Die Datei soll über einen Cache bearbeitet werden (PERFORMANCE=*HIGH). Diese Angabe wird wirksam, da für die Benutzerkennung im Benutzerkatalog die Berechtigung CONCURRENT eingetragen ist.
- Die Anforderung der hochperformanten Dateiverarbeitung soll für Lese- und Schreiboperationen gelten (USAGE=*READ-WRITE).
- Mit DISK-WRITE=*BY-CLOSE vereinbart der Benutzer, dass die Daten erst zum Zeitpunkt des Schließens der Datei in einem konsistenten Zustand gespeichert werden. Daher kann die Datei in einem Cache im Hauptspeicher verarbeitet werden. Mit diesem Attribut sollten nur solche Dateien versehen werden, deren Daten einfach wieder herzustellen sind (z.B. List-Datei bei einer Compilierung), da ein Systemfehler zu Inkonsistenzen dieser Dateien führen kann.

5.7.6 Das HIPERBATCH-Konzept

Mit HIPERBATCH (High Performance Batch Processing) wird die Nutzung einer speziellen neuen Variante des PFA-Cachings bezeichnet.

Bei einer Batch-Verarbeitung liegt oft eine Folge mehrerer Verarbeitungsschritte für einzelne Dateien vor. So wird z. B. in einem Verarbeitungsschritt eine (temporäre) Datei erzeugt die in einem folgenden Verarbeitungsschritt als Eingabe-Datei wieder gelesen und weiterbenutzt wird. Zwischen zwei solchen Verarbeitungsschritten wird die Datei in der Regel geschlossen und wieder geöffnet.

Nach dem Schließen einer mit PFA zwischengepufferten Datei werden

- die zur Datei gehörenden Daten im Cache auf die Platte zurückgeschrieben (im Falle eines Schreib-Cache),
- die Cache-Verwaltungsdaten dieser Datei vom Software-Cache-Handler DAB (d.h. Cache im Hauptspeicher) freigegeben,
- die noch im Cache befindlichen und zu dieser Datei gehörenden Daten invalidiert.

Solange keine Folgeverarbeitung für diese Datei stattfindet, ist dies eine optimierte Vorgehensweise. Bei einem nachfolgenden Zugriff muss die Datei allerdings erst wieder in einer „Einschwingphase“ in den Cache eingelagert werden, bis die Anwendung von Read Hits profitiert. Das vorangegangene Zurückschreiben auf Platte ist bei einer Batch-Verarbeitung hinsichtlich der Datensicherheit nicht nötig, da der Lauf im Fehlerfall wiederholbar ist.

Hier setzt das HIPERBATCH-Konzept an. Über einen neuen CLOSE-Parameter (Kommando ADD-FILE-LINK oder Makro CLOSE) kann eingestellt werden, dass zum CLOSE-Zeitpunkt die im Cache befindlichen Daten nicht zurückgeschrieben und insbesondere nicht invalidiert werden. Ein nachfolgender OPEN auf die gleiche Datei kann die Daten im Cache sofort nutzen. Der Effekt ist eine spürbare Beschleunigung von Batch-Prozessen mit Datei-Folgeverarbeitungsschritten.

Der CLOSE-Parameter kann folgendermaßen angegeben werden:

- Kommando: ADD-FILE-LINK ...,CLOSE-MODE=*KEEP-DATA-IN-CACHE
- Makro: CLOSE <fcb>,*KEEP-DATA-IN-CACHE

6 Datei- und Datenschutz

Dieses Kapitel beschreibt die Funktionen, mit denen das DVS Datei- und Datenschutz unterstützt. Datei- und Datenschutz sind über verschiedene Mechanismen realisiert:
Der Schutz vor unberechtigtem Lesen, Ändern und Zerstören einer bestehenden Datei wird durch Einschränkung der Zugriffsberechtigung und Zugriffsart, durch Vergabe einer Schutzfrist, Kennwörter usw. gewährleistet.
Beim Löschen von Dateien können deren Daten mit X'00' überschrieben werden. Dieser Vorgang wird als „physikalisch löschen“ bezeichnet.
Schutz vor unbefugtem Zugriff ist standardmäßig gewährleistet. Geben Sie die Datei auch für Fremdbenutzer frei, können Sie deren Zugriffsrechte u.a. durch Kennwortvergabe einschränken.

Durch die Dateiverschlüsselung mit Crypto-Kennwort ist es möglich, den Inhalt einer Datei vor jedem unbefugten Zugriff zu schützen – auch gegenüber Personen mit TSOS-Privileg. Dateiverschlüsselung beinhaltet aber keinen Schutz gegen Löschen, Überschreiben oder Zerstören des Dateiinhalts und kann Dateischutz und Sicherung nicht ersetzen.

Für alle Datenträgertypen gewährleistet das DVS vollen Dateischutz (bei Bändern nur, soweit sie mit Standardkennsätzen erstellt wurden). Bei Pubsets beginnt der Zugriffsschutz bereits mit der Erteilung der Zugriffsberechtigung zu einem Pubset, die von der Systembetreuung im Benutzerkatalog-Eintrag definiert wird. Nur Benutzer, denen der Zugriff auf ein Pubset erlaubt ist, können auch auf Dateien dieses Pubsets zugreifen.
Auf Dateiebene kann der Benutzer selber Zugriffsrechte vergeben.

Jeder Zugriff auf eine Datei (Inhaltsanzeige, Bearbeiten mit einem Editor, Ausführen der Kommandos in einer Prozedurdatei,...) lässt sich einer der drei folgenden Zugriffsarten zuordnen:

READ	Lesender Zugriff
WRITE	Schreibender Zugriff
EXECUTE	Ausführender Zugriff

Ein entsprechend vergebener Zugriffschutz kann so zur Wirkung kommen.

6.1 Schutzmechanismen

Im BS2000 existieren für Dateien folgende Schutzmechanismen:

- Begrenzter Pubset-Zugriff
  Durch Verteilung von Benutzerkennungen auf unterschiedliche Pubsets kann ein Schutz vor unerlaubtem Zugriff auf Dateien jeweils anderer Pubsets erreicht werden.
- Standard-Zugriffskontrolle (USER-ACCESS/SHARE und ACCESS) (siehe "Standard-Zugriffskontrolle (USER-ACCESS/SHARE und ACCESS)")
- Einfache Zugriffskontroll-Liste (Basic Access Control List, BACL) Permanente Dateien und Dateigenerationen auf gemeinschaftlichen Datenträgern können durch eine Zugriffskontroll-Liste geschützt werden, Dateien auf Bändern und temporäre Dateien nicht (siehe "Einfache Zugriffskontroll-Liste BACL").
- Definieren von Zugriffsprofilen mittels GUARDS
  GUARDS ist Bestandteil des Software-Produkts SECOS (siehe "GUARDS - Dateischutz über ein spezielles Zugriffsprofil (Guard)").
- Kennwort
  Für jede Datei können Kennwörter (Lese-, Schreib- und Ausführungs-Kennwort) vereinbart werden. Vor der Verarbeitung kennwortgeschützter Dateien ist das entsprechende Kennwort anzugeben. Kennwörter können auf Wunsch verschlüsselt werden(siehe "Vergabe eines Dateikennwortes").
- Dateiverschlüsselung
  Der Inhalt von Dateien wird verschlüsselt (siehe "Dateiverschlüsselung").
- Frist
  Einer Datei kann eine Schutzfrist zugeordnet werden, innerhalb der diese Datei nicht geändert werden darf (siehe "Vergabe einer Schutzfrist").

Hinweise

- Eine mit GUARDS geschützte Datei ist nur zugreifbar, wenn das vereinbarte Zugriffsprofil existiert und das Subsystem GUARDS geladen ist. Über geladene Subsysteme informiert das Kommando SHOW-SUBSYSTEM-STATUS.
- Für Dateien auf Net-Storage und insbesondere für Node-Files ist hinsichtlich ihres Zugriffsschutzes zu beachten, dass die hier beschriebenen Mechanismen nur innerhalb des BS2000 wirken. Beachten Sie außerdem bitte die Hinweise im Abschnitt „Randbedingungen“.

6.1.1 Hierarchie der Schutzmechanismen

Jede Datei kann durch einen oder mehrere Schutzmechanismen gesichert werden.

Beim gleichzeitigen Einsatz der Schutzmechanismen ACCESS/USER-ACCESS, BACL und GUARDS für dasselbe Objekt wird gemäß der folgenden Hierarchie nur einer wirksam:

1. Guards
   Wenn der Schutz eines Objektes über Guards definiert ist, dann gelten ausschließlich die in den Guards festgelegten Zugriffsbedingungen. Eine ggf. für das Objekt definierte BACL und die Schutzattribute ACCESS /USER-ACCESS bleiben unberücksichtigt.
2. einfache Zugriffskontroll-Liste BACL
   Wenn kein Guardschutz für ein Objekt definiert ist, aber eine BACL, dann gilt die in der BACL festgelegte Schutzeinstellung. Die Schutzattribute ACCESS und USER-ACCESS bleiben unberücksichtigt.
3. Standard-Zugriffskontrolle (USER-ACCESS/SHARE und ACCESS)
   Wenn der Schutz eines Objektes weder mit Guards noch mit BACL geregelt ist, dann werden für den Schutzmechanismus die Schutzattribute ACCESS und USER-ACCESS herangezogen.

In allen Fällen wirken zusätzlich der begrenzte Pubset-Zugriff, der Kennwortschutz, die Dateiverschlüsselung und die Schutzfrist.

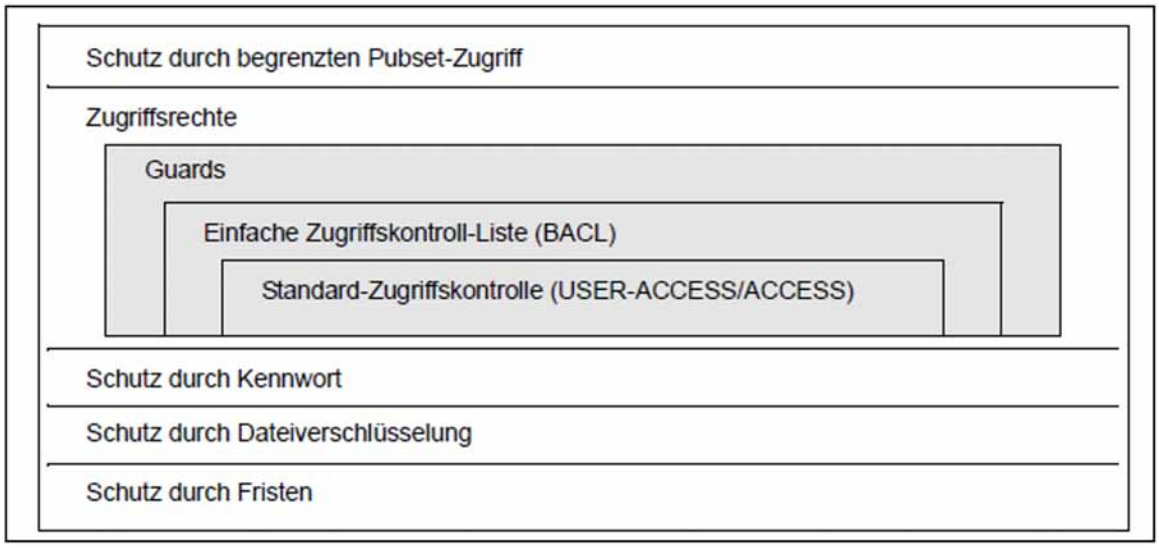


Bild 6: Schutzmechanismen für Dateien auf einem Pubset

6.1.2 Abhängigkeiten der verschiedenen Schutzmechanismen

Standard-Zugriffskontrolle (USER-ACCESS/SHARE und ACCESS)

Bei der Erstellung des Katalogeintrags (Makro CATAL; Kommando CREATE-FILE) wird standardmäßig Dateischutz über die Standard-Zugriffskontrolle eingestellt.
Angaben für die Standard-Zugriffskontrolle werden immer im Dateikatalog eingetragen, unabhängig davon, ob BACL eingeschaltet ist oder ob für die Datei ein GUARDS-Eintrag existiert. Ist zugleich ein höherer Zugriffsschutz vereinbart (BACL oder GUARDS), werden die Werte der Standard-Zugriffskontrolle zwar in den Dateikatalog eingetragen, für diesen Zugriff jedoch ignoriert.

Einfache Zugriffskontroll-Liste BACL

Den Dateischutz über die einfache Zugriffskontroll-Liste (BACL) muss der Benutzer selbst aktivieren (Kommandos CREATE-FILE oder MODIFY-FILE-ATTRIBUTES; Makro CATAL). Angaben für die BACL werden im Dateikatalog eingetragen. Ist gleichzeitig mit BACL ein höherer Zugriffsschutz vereinbart (GUARDS), werden die Werte von BACL beim Zugriff ignoriert.

Wird eine vorhandene einfache Zugriffskontroll-Liste (BACL) gelöscht (Kommando MODIFY-FILE-ATTRIBUTES; Makro CATAL) und existiert kein höherer Zugriffsschutz, werden zuvor oder parallel vergebene Werte für die Standard-Zugriffskontrolle wieder aktiviert, d.h. die Schutzfunktion Standard-Zugriffskontrolle wird wieder wirksam.

GUARDS

Besitzt eine Datei einen GUARDS-Eintrag, wird bei der Zugriffskontrolle nur der GUARDS-Schutz ausgewertet.

Bei der Deaktivierung von GUARDS wird ein ggf. bestehender BACL-Schutz wieder wirksam.

Ist kein BACL-Schutz vereinbart, werden nach Deaktivierung von GUARDS die Schutzattribute USER-ACCESS /ACCESS wieder wirksam.

6.1.3 Schutzmechanismen für Platten- und Banddateien

Die folgende Tabelle gibt einen Überblick über die möglichen Schutzmechanismen:

Objekt	Schutzmechanismus	1. Begrenzter Pubset-Zugriff 2. ACCESS, USER-ACCESS 3. BACL 4. Kennwort 5. Dateiverschlüsselung 6. Schutzfrist 7. GUARDS						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Datei	public	+	+	+	+	+	+	+
	temporär	-	-	-	-	+	-	-
	privat	-	+	+	+	-	+	-
	Band	-	+	-	+	-	+	-
Dateigenerationsgruppe	Index public, FGen public	-	+	+	+	+	+	+
	Index public, FGen Band	-	+	+	+	+	+	+
	Index privat, FGen privat	-	+	+	+	-	+	-

Tabelle 15: Einsatzmöglichkeiten der Schutzmechanismen (+ =anwendbar, - =nicht anwendbar)

Der Gültigkeitsbereich der Schutzmechanismen unterscheidet sich bei Platten- und Banddateien nach Dateitypen: Die Attribute zu „Standard-Zugriffskontrolle“, „Schutzfrist“, „Kennwort“ und „DESTROY-Option“ gelten in vollem Umfang oder nur teilweise.

Plattendateien

Die folgenden Tabellen geben den Wertebereich für die jeweiligen Operanden im Makro bzw. im Kommando an.

Makro

Dateityp	Standard-Zugriffskontrolle		BACL	GUARDS	RETPD	RDPASS, WRPASS, EXPASS	DESTROY
	ACCESS	SHARE					
Permanent	Y	Y	Y	Y	Y	Y	Y
Temporär	WRITE	NO	N	N	wird ignoriert	NONE	Y
Generation	Y	Y	Y	Y	Y	READ/ WRITE	Y

Tabelle 16: Schutzmechanismen für Plattendateien (Makro)

Kommando

Dateityp	Standard-Zugriffskontrolle		BACL	GUARDS	EXPIRATION-DATE	READ-, WRITE-, EXEC-PASSWORD	DESTROY
	ACCESS	SHARE					
Permanent	Y	Y	Y	Y	Y	Y	Y
Temporär	WRITE	OWNER-ONLY	N	N	wird abgewiesen	*NONE	Y
Generation	Y	Y	Y	Y	Y	READ/ WRITE	Y

Tabelle 17: Schutzmechanismen für Plattendateien (Kommando)

Banddateien

Die folgenden Tabellen geben den Wertebereich für die jeweiligen Operanden im Makro und im Kommando an.

Makro

Dateityp	Standard-Zugriffskontrolle		RETPD	RDPASS, WRPASS, EXPASS	DESTROY
	ACCESS	SHARE			
Permanent	Y	Y	Y	Y	Y
Temporär	Y	Y	wird ignoriert	Y	Y
Generation	Y	Y	Y	READ/ WRITE	Y

Tabelle 18: Schutzmechanismen für Banddateien (Makro)

Kommando

Dateityp	Standard-Zugriffskontrolle		EXPIRATION-DATE	READ-, WRITE-, EXEC-PASSWORD	DESTROY
	ACCESS	SHARE			
Permanent	Y	Y	Y	Y	Y
Temporär	Y	Y	N	Y	Y
Generation	Y	Y	Y	READ/ WRITE	Y

Tabelle 19: Schutzmechanismen für Banddateien (Kommando)

Y:	Operand mit allen möglichen Operandenwerten zugelassen
N:	Operand nicht zugelassen
andere Einträge:	diese Einträge sind Standardwerte, die nicht verändert werden können

6.2 Festlegung von Schutzmerkmalen

Wenn eine Datei mit dem Makro CATAL bzw. mit dem Kommando CREATE-FILE katalogisiert wird, ohne dass explizit Schutzmerkmale definiert werden, kann standardmäßig nur der Dateieigentümer auf die Datei zugreifen.

Als **Dateieigentümer** gilt der Benutzer, unter dessen Benutzerkennung die Datei katalogisiert ist, evtl. festgelegte Miteigentümer der Datei (siehe "Festlegung einer Miteigentümerschaft (Co-Owner)") sowie – mit Einschränkungen – die Systembetreuung (Kennungen mit dem Privileg TSOS, siehe "Einschränkungen der TSOS-Miteigentümerschaft").

Standardmäßig ist als Zugriffsart Schreibzugriff (ACCESS=WRITE) eingestellt.
Wird eine Dateigeneration katalogisiert, übernimmt das DVS die im Gruppeneintrag definierten Schutzmerkmale (Dateigenerationen: siehe "Dateigenerationsgruppen (FGG)").

Kennwörter oder Schutzfristen werden vom DVS nicht automatisch definiert, diese Schutzmerkmale können nur vom Eigentümer selbst vergeben werden. Auch Schreibzugriff wird nicht standardmäßig unterbunden.

Soll eine Datei kopiert werden, können mit COPY-FILE PROTECTION=*SAME bzw. COPFILE PROTECT=*SAME für die Kopie dieselben Schutzmerkmale festgelegt werden, wie sie die Originaldatei hat, d.h. die gleichen Kennwörter, die gleiche Schutzfrist usw., jedoch nicht das Dateiüberwachungsattribut (Audit) und die Sperren gegen Speicherplatzfreigabe.

Nur die Dateieigentümer (Definition siehe oben) können Zugriffsrechte vergeben bzw. Dateischutzmerkmale festlegen, ändern oder aufheben.
Für Plattendateien können die Schutzmerkmale jederzeit mit dem Makro CATAL bzw. mit dem Kommando MODIFY-FILE-ATTRIBUTES neu definiert werden, falls die Datei nicht durch einen anderen Zugriff gesperrt ist. Für Banddateien lassen sie sich nur vor dem ersten Eröffnen der Datei ändern. Bei der Dateiverarbeitung werden sie in die Kennsätze übernommen und können dann nicht mehr verändert werden.

Für jede Datei kann sowohl der Kreis der Mitbenutzer als auch die Art des Zugriffs eingeschränkt werden (Standard-Zugriffskontrolle, BACL oder GUARDS).

Zugriffsberechtigung für Dateien

Die Systembetreuung kann den Zugriff zu einem Pubset einschränken: es dürfen nur die Benutzer auf einen Pubset zugreifen, denen explizit durch Eintrag im Benutzerkatalog die Zugriffsberechtigung erteilt wurde (Kommandos ADD-USER oder MODIFY-USER-ATTRIBUTES).

Benutzer können den Zugriff auf eine Datei auf bestimmte Benutzerklassen (OWNER, GROUP, OTHERS) beschränken. Die Definition von Benutzergruppen ist erst bei Einsatz des Software-Produkts SECOS möglich.

Ist SECOS noch nicht eingesetzt, werden vorhandene Einträge in der Benutzerklasse GROUP ignoriert und nur die Einträge der Benutzerklassen OWNER und OTHERS ausgewertet.

Hinweis zur Benutzerklasse GROUP

Alle Benutzer, die keiner explizit eingerichteten Gruppe zugeordnet sind, sind automatisch Mitglied der implizit eingerichteten Gruppe *UNIVERSAL. Dies gilt insbesondere dann, wenn gar keine Gruppen explizit eingerichtet wurden. In diesem Fall sind alle Systembenutzer Mitglied derselben Gruppe. Bei der Auswertung einer BACL erhalten daher alle zugreifenden Benutzerkennungen außer dem Objekteigentümer selbst die Zugriffsrechte aus dem GROUP-Eintrag und nicht die des OTHERS-Eintrags.

Für Mitglieder der Benutzergruppe *UNIVERSAL wird daher dringend empfohlen, für die Benutzerklassen GROUP und OTHERS die gleichen Zugriffsrechte zu vergeben.

6.2.1 Standard-Zugriffskontrolle (USER-ACCESS/SHARE und ACCESS)

Auf eine Datei kann in Abhängigkeit von der Zugriffsart und der Zugriffsberechtigung zugegriffen werden.

Mit dem Operanden ACCESS des Makros CATAL bzw. der Kommandos CREATE-FILE, MODIFY-FILE-ATTRIBUTES, CREATE-FILE-GROUP und MODIFY-FILE-GROUP-ATTRIBUTES kann ein Benutzer festlegen, welchen Zugriff er auf seine Datei erlaubt.

ACCESS=WRITE	Lese- und Schreibzugriffe sind erlaubt.
ACCESS=READ	Nur Lesezugriff ist erlaubt.

Die Berechtigung zur Zugriffsart „Ausführen“ kann bei Verwendung der Standard-Zugriffskontrolle nur mittelbar über die Vergabe eines EXECUTE-Kennworts eingeschränkt werden (siehe Abschnitt „Zugriff zu kennwortgeschützten Dateien“).

Im Makro CATAL wird mit dem Operanden SHARE festgelegt, welche Benutzerkennungen auf die Datei mit den zuvor vergebenen Zugriffstypen zugreifen können:

SHARE=NO	Nur der Dateieigentümer kann zugreifen.
SHARE=YES	Alle Benutzerkennungen dürfen zugreifen, mit Ausnahme der Benutzerkennungen, die das Privileg HARDWARE-MAINTENANCE, aber nicht das Privileg STD-PROCESSING besitzen (siehe Hinweis).
SHARE=SPECIAL	Alle Benutzerkennungen dürfen zugreifen.

In den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES wird mit dem Operanden USER-ACCESS festgelegt, welche Benutzerkennungen auf die Datei mit den zuvor vergebenen Zugriffstypen zugreifen können:

USER-ACCESS=*OWNER-ONLY	Nur der Dateieigentümer kann zugreifen.
USER-ACCESS=*ALL-USERS	Alle Benutzerkennungen dürfen zugreifen, mit Ausnahme der Benutzerkennungen, die das Privileg HARDWARE-MAINTENANCE, aber nicht das Privileg STD-PROCESSING besitzen (siehe Hinweis).
USER-ACCESS=*SPECIAL	Alle Benutzerkennungen dürfen zugreifen.

> ¡ *Hinweis für Benutzerkennungen mit dem Privileg HARDWARE-MAINTENANCE*
>
> Diese Benutzerkennungen zählen standardmäßig **nicht** zur Menge der Benutzer, die mit *ALL-USERS bezeichnet werden.
> Benutzerkennungen mit HARDWARE-MAINTENANCE werden vom Hersteller für die HW-Wartung verwendet, während alle übrigen Benutzerkennungen für den Kunden vorgesehen sind.

6.2.2 Einfache Zugriffskontroll-Liste BACL

Dateien können auch durch die „einfache Zugriffskontroll-Liste“ BACL (Basic Access Control List) geschützt werden.

Nur permanente und Arbeitsdateien auf Platten können durch eine BACL geschützt werden. Temporäre Dateien sowie Banddateien werden durch eine BACL nicht geschützt. Bei der BACL können unabhängig voneinander in beliebiger Kombination die verschiedenen Zugriffsrechte für verschiedene Benutzerklassen vergeben werden. Diese Kombinationen werden in der einfachen Zugriffskontroll-Liste eingetragen.

Die BACL ist für ein Objekt dann wirksam, wenn für das Objekt kein Guards-Schutz definiert ist. Kennwortschutz und Schutzfrist sind zusätzlich wirksam.

Benutzerklassen

Beim Zugriffschutz mit BACL unterscheidet man drei Benutzerklassen:

OWNER: der Eigentümer eines Objekts, also die Benutzerkennung, unter der die Datei katalogisiert ist, sowie Miteigentümer, die mit Hilfe des Miteigentümerschutzes festgelegt wurden (siehe "Festlegung einer Miteigentümerschaft (Co-Owner)").

GROUP: alle Benutzerkennungen der Benutzergruppe, der der Eigentümer angehört, mit Ausnahme des Eigentümers selbst und der Miteigentümer; geprüft wird die Gruppenstruktur des Home-Pubsets.

OTHERS: alle übrigen Benutzer mit Ausnahme der Miteigentümer.

Zugriffsrechte

In einer BACL sind neun Zugriffsberechtigungen für eine Datei festgelegt. Der Datei können für jede der drei Benutzerklassen OWNER, GROUP, OTHERS drei Zugriffstypen separat zugeordnet werden:

- Lesen (R): die Datei darf gelesen und kopiert werden
- Schreiben (W):in die Datei kann geschrieben bzw. sie darf überschrieben werden Anders als beim ACCESS-Attribut beinhaltet das Schreibrecht der BACL nicht automatisch auch das Leserecht (Datenschutz)
- Ausführen (X): die Datei darf ausgeführt werden (Programm, Prozedur)

Keines dieser Zugriffsrechte schließt ein anderes ein.

Jedes der Attribute R, W und X kann gesetzt oder nicht gesetzt sein. Sie gelten unabhängig voneinander und ausschließlich für die angegebene Zugriffsart. Insbesondere ist für „X“ nicht Voraussetzung, dass „R“ gesetzt ist. Durch Angabe von NO-ACCESS für alle Benutzerklassen können dem Eigentümer bzw. den Benutzerkennungen der Gruppe bzw. den Benutzerkennungen außerhalb der Benutzergruppe alle Zugriffsrechte entzogen werden.

Die folgenden Tabellen zeigen, welche Werte für eine Datei in ihre Zugriffskontroll-Liste eingetragen werden müssen, um die gleiche Wirkung zu erzeugen, wie mit den Werten für die Operanden SHARE und ACCESS im Makro CATAL bzw. den Operanden ACCESS und USER-ACCESS im Kommando CREATE-FILE.

Standard-Zugriffskontrolle (CATAL-Makro)		**Einfache Zugriffskontroll-Liste BACL**		
ACCESS	**SHARE**	**OWNER**	**GROUP**	**OTHERS**
WRITE	NO	R W X	- - -	- - -

WRITE	YES	R W X	R W X	R W X
WRITE	SPECIAL	R W X	R W X	R W X
READ	NO	R - X	- - -	- - -
READ	YES	R - X	R - X	R - X
READ	SPECIAL	R - X	R - X	R - X

Tabelle 20: BACL-Werte gemäß Standardzugriffskontroll-Attributen im Makro CATAL

Standard-Zugriffskontrolle (CREATE-FILE-Kommando)		**Einfache Zugriffskontroll-Liste BACL**		
ACCESS	**USER-ACCESS**	**OWNER**	**GROUP**	**OTHERS**
WRITE	OWNER-ONLY	R W X	- - -	- - -
WRITE	ALL-USERS	R W X	R W X	R W X
WRITE	SPECIAL	R W X	R W X	R W X
READ	OWNER-ONLY	R - X	- - -	- - -
READ	ALL-USERS	R - X	R - X	R - X
READ	SPECIAL	R - X	R - X	R - X

Tabelle 21: BACL-Werte gemäß Standardzugriffskontroll-Attributen im Kommando CREATE-FILE

Auswertung der einfachen Zugriffskontroll-Liste

1. Ist die Benutzerkennung, die den Zugriff wünscht, der (Mit-) Eigentümer des Objekts bzw. die Systembetreuung, gelten die unter OWNER abgespeicherten Zugriffsrechte.
2. Gehört die Benutzerkennung der Benutzergruppe des Eigentümers an, gelten die unter GROUP abgespeicherten Zugriffsrechte.
3. Für alle anderen Benutzerkennungen gelten die unter OTHERS abgespeicherten Zugriffsrechte.

Beispiel

OWNER	GROUP	OTHERS
R W X	R W -	R - -

Der (Mit-) Eigentümer dieser Datei darf auf die Datei lesend, schreibend und ausführend zugreifen. Die Gruppe des Dateieigentümers darf die Datei lesen und in die Datei schreiben. Der Rest darf die Datei nur lesen.

Hinweise

Nur bei Einsatz des Software-Produkts SECOS werden die Einträge aller drei Benutzerklassen vollständig ausgewertet (siehe Handbuch „SECOS“ [8]).

Die Definition von Benutzergruppen ist erst bei Einsatz des Software-Produkts SECOS möglich. Ist SECOS noch nicht eingesetzt, werden vorhandene Einträge in der Benutzerklasse GROUP ignoriert und nur die Einträge der Benutzerklassen OWNER und OTHERS ausgewertet.

In Hinblick auf den möglichen Einsatz von SECOS wird dennoch empfohlen, für die Benutzerklasse GROUP die gleichen Einträge aufzunehmen wie für OTHERS.

Aktivierung des BACL-Schutzes

BACL-Schutz wird aktiviert, wenn mindestens für eine Benutzerklasse ein BACL-Eintrag existiert.

In einer einfachen Zugriffskontroll-Liste können die Zugriffstypen READ, WRITE, EXECUTE (kurz R, W und X) in folgender Weise für Benutzerklassen vergeben werden:

Auf Makroebene wird BACL über die Operanden BASACL, OWNERAR, GROUPAR und OTHERAR des Makros CATAL eingestellt. Die Angabe eines Operanden ohne Operandenwert wird ignoriert.

Auf Kommandoebene wird BACL über den Operanden BASIC-ACL bei den Kommandos CREATE-FILE(-GROUP) und MODIFY-FILE(-GROUP)-ATTRIBUTES aktiviert.

Deaktivierung des BACL-Schutzes

Ein bestehender BACL-Schutz kann durch explizite Angabe zurückgesetzt werden (Makro CATAL, Kommando MODIFY-FILE(-GROUP)-ATTRIBUTES).
Wird ein höherer Zugriffsschutz aktiviert (GUARDS), wird der Schutz über BACL ignoriert.

6.2.3 GUARDS - Dateischutz über ein spezielles Zugriffsprofil (Guard)

Die Zugriffskontrolle erfolgt über spezielle Zugriffsprofile, sog. Guards.
Dieser Zugriffsschutz wird nur wirksam, wenn die Funktionseinheit GUARDS (Generally Usable Access Control Administration System) des Software-Produkts SECOS geladen ist (siehe Handbuch „SECOS“ [8]).

Der Zugriff auf eine Datei wird über einen Guard geregelt, der alle Bedingungen enthält, unter denen ein Zugriff erlaubt oder verwehrt wird (Datum, Uhrzeit, Zeitraum, Benutzerkennung). Jedes Zugriffsprofil wird mit entsprechenden GUARDS-Kommandos erstellt und unter einem vom Benutzer vergebenen Guard-Namen im Guard-Katalog als Guard-Eintrag abgespeichert. Jedes Pubset verfügt über einen Guard-Katalog, der unabhängig von den Benutzerdateien verwaltet wird.

Bei der Dateiverarbeitung ist zu unterscheiden zwischen der Verknüpfung der Datei mit einem Guard-Eintrag und dem Zugriff auf eine mit diesem Guard-Eintrag geschützte Datei. Verknüpft wird eine Datei mit einem Guard-Eintrag, indem bei den entsprechenden Operanden des Makros bzw. des Kommandos (CATAL; CREATE-FILE(-GROUP) oder MODIFY-FILE(-GROUP)-ATTRIBUTES) ein Guard-Name eingetragen wird.
Der Zugriff auf eine mit einem Guard geschützte Datei ist nur möglich, falls die im Guard-Eintrag festgelegten Bedingungen dies zulassen.

Aktivierung des GUARDS-Schutzes

GUARDS wird aktiviert, wenn mindestens eine Zugriffsart mit einem Guard-Eintrag verknüpft wird (auch der Operandenwert *NONE bei READ/WRITE/EXEC im Makro bzw. in den Kommandos ist in diesem Sinne ein Guard-Eintrag). Das Zugriffsprofil muss zu diesem Zeitpunkt noch nicht im Guard-Katalog definiert sein. Jede der drei Zugriffsarten (Lesen, Schreiben, Ausführen) kann über einen gesonderten Guard-Eintrag geschützt werden.

> ¡ Anders als beim ACCESS-Attribut impliziert das Schreibrecht bei GUARDS nicht das Leserecht.

Wird für eine Datei ein GUARDS-Schutz aktiviert, werden alle nicht explizit gesetzten Zugriffsarten mit *NONE belegt. Der Zugriff über diese Zugriffsarten ist dann **nicht** möglich.

Erst zum Zeitpunkt des Zugriffs auf eine mit GUARDS geschützte Datei wird geprüft, ob der angegebene Guard-Eintrag (Guard-Name) existiert, ob er verwendet werden darf und ob dem Benutzer auf Grund dieses Zugriffsprofils ein Zugriff in der entsprechenden Zugriffsart erlaubt ist.

Auch Miteigentümer haben für eine von Guards geschützte Datei nur die im Guard-Eintrag definierten Zugriffsrechte.

> ¡ Auf eine Datei kann nicht zugegriffen werden, wenn im Dateikatalog Schutz über GUARDS eingetragen ist, für den angegebenen Guard-Namen jedoch kein Zugriffsprofil im Guard-Katalog definiert ist (z.B. wenn das Zugriffprofil gelöscht wurde). Der Zugriff auf mit GUARDS geschützte Dateien ist grundsätzlich nur möglich, wenn das Subsystem GUARDS geladen ist.

Deaktivierung des GUARDS-Schutzes

Ein bestehender GUARDS-Schutz kann nur durch explizite Angabe zurückgesetzt werden (Makro CATAL, Operand GUARDS=NONE; Kommando MODIFY-FILE(-GROUP)-ATTRIBUTES, Operand GUARDS=NONE).

6.2.4 Vergabe eines Dateikennwortes

Dateien können mit Kennwörtern geschützt werden. Dabei ist es möglich, für jede Zugriffsart ein eigenes Kennwort zu vergeben. Kennwörter werden bei der Erstellung oder der Änderung des Katalogeintrags vergeben (z.B. mit den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES, Operanden READ-/WRITE-/EXEC-PASSWORD oder beim Makro CATAL, Operanden RDPASS, WRPASS und EXPASS).

Es gibt Lese-, Schreib- und Ausführungskennwörter. Die Kenntnis eines Schreibkennworts berechtigt auch zum Lesen und Ausführen der Datei, die Kenntnis des Lesekennworts berechtigt auch zum Ausführen.
Das DVS lässt Dateizugriff nur dann zu, wenn im anfordernden Auftrag das entsprechende Kennwort angegeben wurde (Kommando ADD-PASSWORD).

Ist kein Schreibzugriff erlaubt (es wurde ein Schreibkennwort vergeben oder es ist nur Lese- oder Ausführungszugriff gestattet), verhindert das DVS das Überschreiben, Erweitern und Löschen der Datei. Die Datei lässt sich nur als Eingabedatei verwenden.
Das für den Schreibzugriff notwendige Kennwort muss vor einer Änderung des Katalogeintrags mit dem Kommando ADD-PASSWORD in die Kennworttabelle des Auftrags eingetragen werden (siehe "Zugriff zu kennwortgeschützten Dateien").

Kennwörter treten in Ablaufprotokollen o.Ä. nicht im Klartext auf!

> **Hinweis**
>
> Wenn eine mit Lese- und/oder Ausführungskennwort geschützte Datei in eine verschlüsselte Datei umgewandelt wird, so verliert die Datei den Schutz durch Lese- und/oder Ausführungskennwort (siehe auch Abschnitt „Dateiverschlüsselung“).

6.2.5 Vergabe einer Schutzfrist

Für Dateien kann eine Schutzfrist festlegt werden, während der das DVS Überschreiben, Erweitern oder Löschen der Datei verhindert. Die Datei lässt sich während der angegebenen Frist nur als Eingabedatei verwenden.
Eine Schutzfrist wird mit den Makros FILE und CATAL, Operand RETPD oder EXDATE, bzw. mit den Kommandos ADD-FILE-LINK und MODIFY-FILE-ATTRIBUTES, Operand EXPIRATION-DATE, vergeben oder nachträglich geändert.

6.2.6 Festlegung eines Löschzeitpunktes

Mit dem für SF-und SM-Pubset-Dateien eingeführten Attribut FREE-FOR-DELETION (siehe Kommandos CREATE-FILE, MODIFY-FILE-ATTRIBUTES bzw. Makro CATAL, Operand DELDATE) hat der Benutzer die Möglichkeit, seinen Dateien eine Frist zuzuordnen, nach der sie – ohne Berücksichtigung des Zugriffsschutzes – gelöscht werden können.

Dateien mit abgelaufener Frist werden durch die Selektionskriterien von SHOW-FILE-ATTRIBUTES und DELETE-FILE bzw. FSTAT und ERASE erfasst. Mit vom Benutzer oder Systembetreuer erstellten Prozeduren können auf einfache Weise Aufräum-Aktionen zum Löschen freigebener Dateien veranlasst werden.
Solche Aufräum-Aktionen können sowohl dezentral durch die jeweiligen Benutzer, als auch zentral durch den Systembetreuer durchgeführt werden. Eine zentral durchgeführte Aufräum-Aktion kann z.B. zur Gewinnung von freiem Speicherplatz bei Speicher-Engpässen beitragen: Sie erlaubt dem Systembetreuer das Löschen freigegebener Dateien.

6.2.7 Beispiele zum Festlegen von Schutzmerkmalen

Makros

Ein Benutzer mit der Benutzerkennung EINERLEI legt in einem Assembler-Programm Schutzmerkmale für verschiedene Dateien mit dem Makro CATAL fest:

```
CATAL  DAT1,...,STATE=*UPDATE,SHARE=*NO,ACCESS=*READ ————————————————  (1)
CATAL  DAT2,...,STATE=*UPDATE,SHARE=*YES,ACCESS=*READ ————————————————  (2)
CATAL  DAT3,...,STATE=*UPDATE,SHARE=*YES,WRPASS=C'007' ———————————————  (3)
CATAL  DAT4,...,STATE=*UPDATE,SHARE=*YES,RDPASS=C'0815',DELDATE='+150'   (4)
CATAL  DAT5,...,STATE=*UPDATE,SHARE=*YES,RDPASS=C'1111',EXPASS=C'2222',
       RETPD=20 ——————————————————————————————————————————————————————  (5)
CATAL  DAT6,...,STATE=*NEW,PROTECT=(*FROM-FILE,DAT5),DESTROY=*YES,
       BASACL=*STD ———————————————————————————————————————————————————  (6)
```

Zur Bedeutung der Programmzeilen siehe bei „Kommandos“.

Auch mit dem Makro FILE können diese Schutzattribute gesetzt oder geändert werden. Die Schutzattribute werden auch durch die Selektionskriterien von FSTAT und ERASE erfasst.

Kommandos

Ein Benutzer mit der Benutzerkennung EINERLEI legt (in einer Prozedur oder im Dialog) Schutzmerkmale für verschiedene Dateien fest.

```
/MODIFY-FILE-ATTRIBUTES DAT1,...,USER-ACCESS=*OWNER-ONLY,ACCESS=*READ –  (1)
/MODIFY-FILE-ATTRIBUTES DAT2,... USER-ACCESS=*ALL-USERS,ACCESS=*READ ——  (2)
/MODIFY-FILE-ATTRIBUTES DAT3,...,USER-ACCESS=*ALL-USERS,
         WRITE-PASSWORD=C'007'————————————————————————————————————————  (3)
/MODIFY-FILE-ATTRIBUTES DAT4,...,USER-ACCESS=*ALL-USERS,
         READ-PASSWORD=C'0815',FREE-FOR-DELETION=+150————————————————  (4)
/MODIFY-FILE-ATTRIBUTES DAT5,...,USER-ACCESS=*ALL-USERS,
         READ-PASSWORD=C'1111',EXECUTE-PASSWORD=C'2222',
         EXPIRATION-DATE=+20————————————————————————————————————————  (5)
CREATE-FILE DAT6,...,PROTECTION-ATTR=*FROM-FILE(FILE-NAME=DAT5),
         DESTROY-BY-DELETE=*YES,BASIC-ACL=*STD——————————————————————  (6)
```

Die Schutzattribute können auch mit den Kommandos CREATE-FILE und ADD-FILE-LINK gesetzt werden und werden auch durch die Selektionskriterien von SHOW-FILE-ATTRIBUTES und DELETE-FILE erfasst.

(1) Nur Benutzer, die unter der Benutzerkennung EINERLEI oder unter der Kennung eines Miteigentümers oder unter TSOS (Einschränkungen siehe "Einschränkungen der TSOS-Miteigentümerschaft") arbeiten, können auf die Datei DAT1 zugreifen. Es ist allerdings kein Schreibzugriff möglich.

(2) Die Datei DAT2 ist mehrbenutzbar, d.h. Dateizugriffe von Aufträgen unter fremden Benutzerkennungen sind möglich. Schreibzugriff wird jedoch durch ACCESS= *READ unterbunden.

(3) Die Datei DAT3 ist ebenfalls mehrbenutzbar.
Schreibzugriff ist möglich, jedoch nur, wenn im anfordernden Auftrag das Kennwort C'007' angegeben wird.

(4) Auch die Datei DAT4 ist mehrbenutzbar. Wenn DAT4 ein ablauffähiges Programm oder eine Prozedur enthält, können beliebige Benutzerkennungen dies(e) ablaufen lassen.
Lese- und Schreibzugriff auf DAT4 ist nur möglich, wenn das Kennwort C'0815' angegeben wird.
Mit DELDATE bzw. FREE-FOR-DELETION wird für die Datei ein Löschzeitpunkt (heute in 150 Tagen) festgelegt, nach dem sie – ohne Berücksichtigung des Zugriffsschutzes – gelöscht werden kann.

(5) Die Datei DAT5 ist mehrbenutzbar, aber ohne Kennwortangabe ist kein Dateizugriff möglich.
Die Eingabe des Ausführungskennwortes C'2222' ermöglicht den Aufruf der Datei mit Kommandos wie START-EXEC-PROGRAM oder CALL-PROCEDURE (Programm- oder Prozedurdatei).
Wird das Lesekennwort C'1111' eingegeben, kann der Benutzer Programm oder Prozedur ablaufen lassen und die Datei lesen.
Nach Ablauf der Schutzfrist (sie beträgt 20 Tage) kann er die Datei unter Angabe des Lesekennwortes verändern oder löschen.

(6) Die neu zu katalogisierende Datei DAT6 soll die Schutzmerkmale der Datei DAT5 übernehmen.
Bei Speicherplatzfreigabe soll die Datei jedoch mit binären Nullen überschrieben werden, unabhängig vom eingestellten Wert bei DAT5.
Die Zugriffskontrolle durch BASIC-ACL wird aktiviert. Beim Neuerstellen eines Katalogeintrags werden für *STD folgende Werte gesetzt: Der Eigentümer bekommt automatisch alle Rechte (Lesen, Schreiben, Ausführen), Gruppenmitglieder und „Andere“ keine Rechte zugeteilt.

6.3 Festlegung einer Miteigentümerschaft (Co-Owner)

Mit Hilfe von SECOS kann ein Objekteigentümer festlegen, für welche seiner Objekte er Miteigentümer benennen will und welche Zugriffsbedingungen die Miteigentümer bei Verwaltungszugriffen zu erfüllen haben. Objekteigentümer ist diejenige Benutzerkennung, unter der das Objekt abgelegt ist. Objekte können Dateien, Jobvariablen oder Bibliotheken sein.

Ein Miteigentümer ist eine Benutzerkennung, die ungleich der des Objekteigentümers ist, aber in Bezug auf ein bestimmtes Objekt dieselben Rechte besitzt wie der Objekteigentümer.

Allgemein gilt für Miteigentümer: Alle Lese-, Schreib- und Ausführungszugriffe auf Dateien werden nach Regeln der traditionellen Dateischutzmechanismen kontrolliert:

- Ist eine Datei über SHARE/ACCESS oder über BASIC-ACL geschützt, hat ein Miteigentümer dieselben Lese-, Schreib- und Ausführungsrechte wie der Dateieigentümer.
- Ist eine Datei durch GUARDS geschützt, erfolgt die Zugriffsregelung durch Auswertung von Zugriffsbedingungen, die in STDAC-Guards festgelegt sind.
- Ist eine Datei mit Crypto-Kennwort verschlüsselt, kann der Zugriff auf den Inhalt erst nach Eingabe des Crypto-Kennworts erfolgen.

Miteigentümer können über die Kommandos

- ADD-/MODIFY-/REMOVE-/SHOW-COOWNER-PROTECTION-RULE und
- ADD-/MODIFY-/REMOVE-/SHOW-ACCESS-CONDITIONS

festgelegt, angezeigt und entfernt werden.

Erzeugt ein Miteigentümer unter fremder Kennung eine Datei und schützt diese mit einem STDAC-Guard, so muss er vor einem Dateizugriff dafür sorgen, dass seiner Kennung ein Zugriffsrecht auf die Datei eingeräumt ist. Umgekehrt muss einem Dateieigentümer bewusst sein, dass ihm Datenzugriffe durch Miteigentümer untersagt werden können.

Mit dem Kommando MODIFY-FILE-ATTRIBUTES können sich jedoch sowohl Dateieigentümer als auch Miteigentümer das Zugriffsrecht jederzeit und uneingeschränkt wieder beschaffen.

6.3.1 Einschränkungen der TSOS-Miteigentümerschaft

Standardmäßig besitzt die Benutzerkennung TSOS systemweit ein uneingeschränktes Mitverwaltungsrecht für Dateien und Jobvariablen. Mit dem Einsatz von SECOS ist es jedoch jedem Benutzer möglich, dieses Recht einzuschränken, wenn bestimmte systemspezifische Voraussetzungen seitens der Systembetreuung geschaffen wurden.

Die Einschränkung der TSOS-Miteigentümerschaft von Dateien wirkt sich auf bestimmte Kommandos und Makros aus, und dort gegebenenfalls nur auf einzelne Operanden. In der folgenden Tabelle sind diese Kommandos und Makros aufgelistet:

Kommandos	**Makros**
MODIFY-FILE-ATTRIBUTES	CATAL (STATE=*UPDATE)
MODIFY-FILE-GENERATION-SUPPORT	CATAL (STATE=*UPDATE)
MODIFY-FILE-GROUP-ATTRIBUTES	CATAL (STATE=*UPDATE)
DELETE-FILE	ERASE
COPY-FILE	COPFILE

Um eine Datei wirksam vor TSOS zu schützen, muss der Objekteigentümer **zwei** benutzerspezifische Schutzeinstellungen vornehmen:

1. Er muss dem Benutzer TSOS das **Mitverwaltungsrecht** für seine Objekte entziehen.
2. Er muss dem Benutzer TSOS das **Zugriffsrecht** für seine Objekte entziehen. Hierzu muss der GUARDS-Zugriffsschutz genutzt werden, da nur mit diesem TSOS-Zugriffe unterbunden werden können.

Es müssen beide Schritte ausgeführt werden, da das Entziehen des Mitverwaltungsrechts im 1. Schritt nur das Ändern von Schutzattributen verbietet. Es verbietet keine Datenzugriffe, wie z.B. das Lesen einer Datei. Dazu muss Schritt 2 ausgeführt werden.

Genaue Angaben über die notwendigen Schritte und die systemspezifischen Einstellungen seitens der Systembetreuung finden Sie im Handbuch „SECOS“ [8].

Eine weitere Möglichkeit, Dateiinhalte vor TSOS-Benutzern zu verbergen, ist das Verschlüsseln der Datei mit einem Crypto-Kennwort (siehe "Dateiverschlüsselung").

6.3.2 Beispiel zur Festlegung von Miteigentümern

USER1 möchte, dass USER2 das Recht hat, unter seiner Kennung (USER1) Dateien anzulegen und zu verwalten, wenn deren Name die Zeichenfolge „TEST“ enthält.

USER1 definiert ein Bedingungsguard COND1, das USER2 zeitlich unbegrenzten Zugriff gewährt.

```
/create-guard cond1,user-inf='Zugriffsbedingungen fuer Coowner'
/add-access-conditions guard-name=cond1, -
/                      subjects=*user(user-identification=user2)
```

Dann definiert USER1 einen Regelbehälter COO1 mit einer Miteigentümerregel. Diese gibt an, dass die Zugriffsbedingungen für Miteigentümer der Dateien, deren Name dem Muster „*TEST*“ entspricht, im Bedingungsguard COND1 festgelegt sind.

```
/create-guard coo1,user-inf='Coowner-Regelbehaelter'

/add-coowner-protection-rule rule-container-guard=coo1, -
/            protection-rule=rule1, -
/            protect-object=*parameters(name=*test*,condition-guard=cond1)
```

Zur Kontolle gibt USER1 Informationen über alle Guards und den Regelbehälter COO1 aus. Voraussetzung: Zu Beginn dieser Beispielsitzung waren keine Guards unter der Kennung USER1 vorhanden.

```
/show-guard-attributes
      Guard name        Scope   Type      Creation Date       LastMod Date
 ------------------------------------------------------------------------------
 :DEL1:$USER1.COND1     USR  STDAC    2011-04-19/10:35:47 2011-04-20/11:36:33
                        Zugriffsbedingungen fuer Coowner
 :DEL1:$USER1.COO1      USR  COOWNERP 2011-04-19/10:37:26 2011-04-20/11:38:53
                        Coowner-Regelbehaelter
 ------------------------------------------------------------------------------
 Guards selected: 2                                              End of display
/show-coowner-protection-rule coo1
 ------------------------------------------------------------------------------
 RULE CONTAINER :DEL1:$USER1.COO1                          COOWNER PROTECTION
 ------------------------------------------------------------------------------
 RULE1          OBJECT     = *TEST*
                CONDITIONS  = $USER1.COND1
                TSOS-ACCESS = SYSTEM-STD
 ------------------------------------------------------------------------------
 RULE CONTAINER SELECTED: 1                                      END OF DISPLAY
```

Da der Name des Regelbehälters nicht den Namenskonventionen für aktive Regelbehälter entspricht, dient er nur zur Vorbereitung der Standardschutzregel. USER2 hat noch keine Miteigentümerberechtigung für Dateien unter der Kennung USER1, wie der Aufruf des Kommandos SHOW-COOWNER-ADMISSION-RULE unter der Kennung USER2 zeigt.

```
/show-coowner-admission-rule $user1.*

 COO3316 NO COOWNER PROTECTION ACTIVE
```

Um den Miteigentümerschutz zu aktivieren, benennt USER1 den inaktiven Regelbehälter COO1 um.

```
/mod-guard-attr guard-name=coo1,new-name=sys.ucf
```

USER1 lässt sich den Inhalt des jetzt aktiven Regelbehälters anzeigen.

```
 /show-coowner-protection-rule
  ------------------------------------------------------------------------------
  RULE CONTAINER :DEL1:$USER1.SYS.UCF                ACTIVE COOWNER PROTECTION
  ------------------------------------------------------------------------------
  RULE1          OBJECT      = *TEST*
                 CONDITIONS  = $USER1.COND1
                 TSOS-ACCESS = SYSTEM-STD
 ------------------------------------------------------------------------------
  RULE CONTAINER SELECTED: 1                                    END OF DISPLAY
```

USER2 überprüft, welche Regeln ihn zum Miteigentümer von Dateien der Kennung USER1 machen.

```
 /show-coowner-admission-rule $user1.*
  ------------------------------------------------------------------------------
  COOWNER RULES FOR FILE  :DEL1:$USER1.*
  ------------------------------------------------------------------------------
  RULE1          OBJECT     = *TEST*
                 CONDITIONS = $USER1.COND1
  ------------------------------------------------------------------------------
  RULES SELECTED: 1                                             END OF DISPLAY
```

Jetzt kann USER2 die Datei TESTTEST unter $USER1 anlegen.

```
 /create-file $user1.testtest
 /show-file-att $user1.testtest
  0000003 :DEL1:$USER1.TESTTEST
  :DEL1: PUBLIC:      1 FILE  RES=        3 FRE=         3 REL=        3 PAGES
```

6.4 Übernahme von Schutzattributen von einer Datei oder über Default-Protection

Die Besetzung von Schutzattributen von Dateien soll oft vorgegebenen Mustern entsprechen. Um dies zu unterstützen, bietet das System die Möglichkeit, sie von existierenden Dateien zu übernehmen oder sie – durch Vorgaben bzgl. des Dateinamens geregelt – mit speziell eingestellten Default-Werten zu belegen.

Mit der SECOS-Funktion „Default-Protection“ (Subsystem GUARDDEF) können pubset- oder benutzerglobale Default-Werte für Schutzattribute festgelegt werden. Diese Default-Werte werden in Attribut-Guards hinterlegt.

Folgende Schutzattribute können mit Default-Protection vorbelegt werden:

Schutzattribut	Bedeutung
ACCESS	Standard-Zugriffskontrolle (Zugriffsart)
USER-ACCESS	Standard-Zugriffskontrolle (Zugriff fremder Benutzer)
BASIC-ACL	Einfache Zugriffskontrolliste
GUARDS	Zugriffskontrolle über GUARDS
READ-PASSWORD	Lesekennwort
WRITE-PASSWORD	Schreibkennwort
EXEC-PASSWORD	Ausführungskennwort
DESTROY-BY-DELETE	Binär Löschen
SPACE-RELEASE-LOCK	Freigabesperre für Speicherplatz
FREE-FOR-DELETION	Freigabedatum zum Löschen
EXPIRATION-DATE	Schutzfrist

Die Werte, die über Default-Protection für einen Dateinamen gültig sein sollen, werden über die Kommandos ADD-/MODIFY-DEFAULT-PROTECTION-RULE und ADD-/MODIFY-DEFAULT-PROTECTION-ATTR festgelegt (siehe Handbuch „SECOS“ [8]).

Vergabe von Schutzattributen

Folgende DVS-Schnittstellen bieten Funktionen zur Übernahme von Schutzattributen an:

- Makro CATAL
- Kommando CREATE-FILE
- Kommando CREATE-FILE-GROUP
- Kommando MODIFY-FILE-ATTRIBUTES
- Kommando MODIFY-FILE-GROUP-ATTRIBUTES

Vergabe von Schutzattributen über Default-Protection

Die Übernahme von Schutzattributen gemäß vordefinierter, namensabhängiger Default-Werte (Default-Protection) kann implizit durch Nichtangabe (Voreinstellung) oder explizit geschehen. Die explizite Übernahme von Schutzattributen erfolgt durch folgende Angaben:

- bei den Kommandos mit dem Operanden PROTECTION=*PAR(PROTECTION-ATTR=*BY-DEF-PROT-OR-STD)
- beim Makro CATAL mit dem Operanden PROTECT=*BY_DEF_PROT_OR_STD

sowie Bezug auf diesen Wert bei den einzelnen Operanden (z.B. ACCESS=*BY-PROTECTION-ATTR).

Vergabe von Schutzattributen von bereits existierenden Dateien

Die Übernahme der Schutzattribute von bereits existierenden Dateien erfolgt:

- bei den Kommandos mit dem Operanden PROTECTION=*PAR(PROTECTION-ATTR=*FROM-FILE(...))
- beim Makro CATAL mit dem Operanden PROTECT=(*FROM-FILE,<dateiname>)

Hinweise

- Bei Übernahme eines Default-Datums wird 0.00 Uhr lokale Zeit eingestellt.
- Für temporäre Dateien können nur die beiden Schutzattribute DESTROY-BY-DELETE und SPACE-RELEASE-LOCK über Default-Protection voreingestellt werden.

Schutzattribute beim Neukatalogisieren von Dateien

Schutzattribut	**PROTECTION-ATTR=**			
	***FROM-FILE**	***STD[1)]**	***BY-DEF-PROT-OR-STD**	
			Default-Protection nicht aktiv[1)]	**Default-Protection aktiv**
ACCESS	von der Referenzdatei übernommener Wert	WRITE	WRITE	von der Default-Protection gelieferter Wert
USER-ACCESS		OWNER-ONLY	OWNER-ONLY	
BASIC-ACL		NONE	NONE	
DESTROY-BY-DELETE		NO	NO	
GUARDS		NONE	NONE	
SPACE-RELEASE-LOCK		NO	NO	
READ-PASSWORD	NONE	NONE	NONE	

WRITE-PASSWORD	NONE	NONE	NONE	
EXEC-PASSWORD	NONE	NONE	NONE	
FREE-FOR-DELETION	NONE	NONE	NONE	NONE
AUDIT	NONE	NONE	NONE	NONE

1) Es werden die System-Standardwerte eingetragen.

Eine Schutzfrist (EXPIRATION-DATE) kann beim Ersteintrag nicht vergeben werden.
Sie ist bei Dateien implizit mit *NONE vorbelegt, bei Dateigenerationsgruppen mit *TODAY.

Schutzattribute beim Ändern von Dateimerkmalen

Schutzattribut	**PROTECTION-ATTR=**				
	***UNCH**	***FROM-FILE**	***STD[1)]**	***BY-DEF-PROT-OR-STD**	
				Default-Protection nicht aktiv[1)]	**Default-Protection aktiv**
ACCESS	UNCHANGED	von der Referenzdatei übernommener Wert	WRITE	WRITE	von der Default-Protection gelieferter Wert
USER-ACCESS	UNCHANGED		OWNER-ONLY	OWNER-ONLY	
BASIC-ACL	UNCHANGED		NONE	NONE	
DESTROY-BY-DELETE	UNCHANGED		NO	NO	
GUARDS	UNCHANGED		NONE	NONE	
SPACE-RELEASE-LOCK	UNCHANGED		NO	NO	
EXPIRATION-DATE 2)	UNCHANGED		TODAY	TODAY	
READ-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
WRITE-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
EXEC-PASSWORD	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
FREE-FOR-DELETION	UNCHANGED	UNCHANGED	UNCHANGED	NONE	
AUDIT	UNCHANGED	UNCHANGED	UNCHANGED	UNCHANGED	UNCHANGED

1) Es werden die System-Standardwerte eingetragen.

2) Die Schutzfrist wird nur dann eingetragen, wenn es sich um eine permanente Datei mit Erstellungsdatum oder um eine Dateigenerationsgruppe handelt. Hat die Referenzdatei keine Schutzfrist, wird *TODAY eingetragen.

Hinweise zur Default-Protection

Default-Protection und Dateitypen

Nicht zu einem Dateityp passende Default-Werte werden ignoriert. Dies betrifft:

- SPACE-RELEASE-LOCK, GUARDS, BASIC-ACL und Lösch-Freigabedatum für Banddateien
- SPACE-RELEASE-LOCK, GUARDS und Lösch-Freigabedatum für Dateien und Dateigenerationsgruppen auf Privatplatten
- GUARDS, BASIC-ACL, Schutzfrist und Lösch-Freigabedatum für temporäre Dateien
- ACCESS, USER-ACCESS und Kennwörter für temporäre Dateien auf Pubsets
- EXEC-Rechte, EXEC-Kennwörter und USER-ACCESS=*SPECIAL für Dateigenerationsgruppen

Für Banddateien mit Erstellungsdatum wird die Angabe PROTECTION-ATTR=*BY-DEF-PROT-OR-STD abgewiesen, da deren Schutzattribute nicht geändert werden können.

Umbenennen von Dateien

Beim Umbenennen von Dateien muss folgendes beachtet werden:

- Nur bei gleichzeitigem Rücksetzen auf Default-Werte wird die Default-Protection für den neuen Dateinamen ausgewertet.
- Beim Umbenennen von permanent nach temporär und umgekehrt mit gleichzeitigem Rücksetzen der Schutzattribute werden die Default-Werte anhand des neuen Dateinamens und -typs bestimmt.
- Beim Umbenennen einer Datei in eine Dateigeneration ist kein gleichzeitiges Rücksetzen der Schutzattribute möglich.

Hierarchie der Schutzfunktionen

Die jeweils eingetragenen Werte werden in dieser Priorität bestimmt:

1. explizite Angabe im Kommando oder Makro
2. von Default-Protection oder über Referenzdatei gelieferter Wert
3. System-Standardwert

Wird ein Schutzattribut explizit angegeben, so wird ein Default-Wert für ein anderes Schutzattribut nicht eingetragen, wenn dieser Wert die explizite Angabe außer Kraft setzen würde. Statt des Default-Wertes wird dann der System-Standardwert eingetragen.

Beispiele

- Ist ACCESS oder USER-ACCESS explizit angegeben, so werden BASIC-ACL und GUARDS nicht besetzt.
- Ist für BASIC-ACL ein Wert ungleich *NONE angegeben, wird GUARDS nicht besetzt.
- Der Wert für das Lösch-Freigabedatum einer Datei wird übergangen, wenn für ACCESS, BASIC-ACL, GUARDS, Kennwörter oder Schutzfrist explizit ein Wertungleich *NONE angegeben wurde.

Kennwörter

Default-Kennwörter werden im Attribut-Guard immer verschlüsselt abgelegt, auch wenn der Systemparameter ENCRYPT den Wert N hat.

Wird ein Default-Kennwort eingetragen, sind anschließend alle Kennwörter der Datei verschlüsselt, ebenfalls unabhängig vom Wert des Systemparameters ENCRYPT.

Beim Neukatalogisieren werden Default-Werte für Kennwörter mit PROTECTION-ATTR= *STD oder PROTECTION-ATTR=*FROM-FILE() nicht eingetragen.

RFA (Remote File Access)

Bei Einsatz von RFA sind immer die am Remote-System gültigen Default-Werte relevant.

Einschränkungen für Default-Protection

Default-Protection kommt unter folgenden Voraussetzungen nicht zum Einsatz:

- beim Importieren einer Datei
- bei Angabe einer einzelnen Dateigeneration
- beim GUARDS-Katalog
- bei Angabe einer Referenzdatei (PROTECTION-ATTR=*FROM-FILE(...))
- bei Angabe von PROTECTION-ATTR=*STD

Einschränkungen beim Neukatalogisieren

Ein Default-Wert für das Lösch-Freigabedatum wird nicht eingetragen.

Die Schutzfrist wird beim Ersteintrag (also bei Dateieröffnung und bei Ersteintrag einer Dateigenerationsgruppe) nicht auf den eingestellten Default-Wert, sondern auf den System-Standardwert gesetzt.

Einschränkungen/Besonderheiten beim Zurücksetzen auf Default-Werte

Es werden (im Gegensatz zum Zurücksetzen auf System-Standardwerte) auch das Lösch-Freigabedatum und die Kennwörter verändert.

Ist der Default-Wert für die Schutzfrist ein bereits vergangenes Datum, wird anstatt des Default-Wertes das aktuelle Tagesdatum eingetragen.

6.5 Weitergabe von Attributen beim Kopieren einer Datei

Beim Kopieren einer Datei mit dem Kommando COPY-FILE bzw. dem Makro COPFILE können Schutzattribute, abhängig vom Wert des Operanden PROTECTION bzw. PROTECT, weitergegeben werden.

Zusätzlich zur bereits bestehenden Möglichkeit, Schutzattribute mit PROTECTION bzw. PROTECT=*SAME weiterzugeben, können Schutzattribute mit PROTECTION bzw. PROTECT=*STD in Verbindung mit der Funktion „Default-Protection“ eingetragen werden.
Mit CHANGE-DATE bzw. CHANGE=*SAME kann das Last-Change-Date von der Quelldatei übernommen werden.

Grundsätzlich werden die Schutzattribute von der Quelldatei nicht übernommen. Dies ist nur beim Kopieren innerhalb der Benutzerkennung des Aufrufers und beim Kopieren einer fremden Datei in eine Datei des Aufrufers möglich. Andernfalls werden Benutzbarkeit und Zugriffserlaubnis mit Standardwerten belegt (also SHARE=NO und ACCESS=WRITE beim Makro COPFILE bzw. USER-ACCESS=*OWNER-ONLY und ACCESS=*WRITE beim Kommando COPY-FILE).

Die Übernahme von Schutzattributen ist abhängig

- von der Richtung des Kopierens
  - innerhalb der gleichen Benutzerkennung
  - außerhalb der eigenen Benutzerkennung
- von der Art, wie die Quelldatei geschützt ist
  - durch Standard-Zugriffskontrolle
  - durch BACL (einfache Zugriffskontroll-Liste)
  - durch GUARDS
- vom Typ der Zieldatei
  - Datei auf einem gemeinschaftlichen Datenträger (Pubset)
  - Datei auf Privatplatte
  - Banddatei
  - Dateigeneration
  - Temporäre Datei

Die Sicherungsattribute (siehe CATAL-Makro) LARGE, BACKUP, MIGRATE, NUM-OF-BACKUP-VERS, OPNBACK und MANCLAS und die Sicherheitsattribute DESTROY, RETPD, ENCRYPT, RDPASS, WRPASS, EXPASS und DELDATE werden unter den o.g. Voraussetzungen kopiert. Bezüglich der Schreibattribute gilt Folgendes:

Kopieren innerhalb der gleichen Benutzerkennung

Quelldatei geschützt durch	Zieldatei auf Pubset	Zieldatei auf Privatplatte	Zieldatei auf Band
Standard-Zugriffskontrolle	(1)	(1)	(1)
einfache Zugriffskontroll-Liste (BACL)	(1)	(1)	(2)
GUARDS	(1)	(3)	(2)

Tabelle 22: Kopieren innerhalb der gleichen Benutzerkennung und CATID

Kopieren von einer fremden Benutzerkennung in die eigene Benutzerkennung

Quelldatei geschützt durch	Zieldatei auf Pubset	Zieldatei auf Privatplatte	Zieldatei auf Band
Standard-Zugriffskontrolle	(1)	(1)	(1)
einfache Zugriffskontroll-Liste (BACL)	(3)	(3)	(2)
GUARDS	(3)	(3)	(2)

Tabelle 23: Kopieren von einer fremden Benutzerkennung oder CATID

(1) Die Attribute der Quelldatei werden kopiert. Falls für die Zieldatei bereits ein Katalogeintrag existiert, werden die darin enthaltenen Attribute ersetzt, ein vorhandener BACL- oder GUARDS-Eintrag wird gelöscht bzw. überschrieben.

(2) Es werden die Standardwerte für Banddateien eingesetzt (SHARE=YES bzw. USER-ACCESS=ALL-USERS, ACCESS=WRITE). Banddateien besitzen keinen BACL- oder GUARDS-Eintrag.

(3) Es werden die Standardwerte für Plattendateien eingesetzt. Falls für die Zieldatei bereits ein Katalogeintrag existiert, werden die darin enthaltenen Attribute bzgl. der Standard-Zugriffskontrolle ersetzt, vorhandene BACL- oder GUARDS-Einträge werden gelöscht.

Kopieren von der eigenen in eine fremde Benutzerkennung

Eine derartige Kopie ist nur möglich, wenn bereits eine Datei unter der fremden Benutzerkennung katalogisiert ist. Es werden nur die Daten in die bestehende Datei kopiert, die Schutzattribute bleiben unverändert. Die Angabe des Operanden PROTECT=SAME im Makro bzw. des Operanden PROTECTION=SAME im Kommando ist hier wirkungslos.

Besonderheiten

- Systembetreuung

  Die Benutzerkennung mit dem Privileg TSOS ist Miteigentümer aller Dateien und darf auch unter fremden Benutzerkennungen neue Dateien einrichten (Einschränkungen siehe "Einschränkungen der TSOS-Miteigentümerschaft"). Beim Kopieren unter einer solchen Benutzerkennung wird daher immer so vorgegangen, wie unter „Kopieren innerhalb der gleichen Benutzerkennung“ beschrieben, auch wenn die Benutzerkennung von Quell- und Zieldatei unterschiedlich ist. Dies gilt nicht für andere Miteigentümer-Kennungen (siehe "Festlegung einer Miteigentümerschaft (Co-Owner)").

- Dateigenerationen

  Für jede Generation einer Generationsgruppe gelten die Schutzattribute, die für die Gruppe definiert sind. Die einzelnen Generationen haben keine separaten Schutzattribute.

  Wird eine Generation kopiert, werden die Schutzattribute der zugehörigen Gruppe benutzt (die ja auch für jede einzelne Generation gelten).

  Wird in eine Gruppe hineinkopiert, erhält die neue Generation die Attribute der Zielgruppe (es werden also keine Attribute der Quelldatei übernommen).

- Temporäre Dateien

  Temporäre Dateien sind task-lokale Dateien. Beim Kopieren einer temporären Datei werden daher keine Schutzattribute der Quelldatei übernommen.

Kopieren mit PROTECTION=*STD bzw. PROTECT=*STD

Ist die Zieldatei nicht vorhanden und Default-Protection aktiv, bekommt die Zieldatei die von der Default-Protection gelieferten Werte der Schutzattribute. Ist Default-Protection nicht aktiv, werden – wie bisher – die System-Standardwerte vergeben.
Ist die Zieldatei vorhanden, bleiben die bestehenden Werte unverändert.

6.6 Berücksichtigung der Schutzattribute bei Dateiverarbeitung

- Löschen, Ändern und Anzeigen von Datei-Attributen
- Zugriff zu kennwortgeschützten Dateien
- Zugriff auf eine mit Schutzfrist geschützte Datei
- Schutz von Mehrdatei-Bändern gegen implizites Löschen
- Übernahme von Schutzattributen in den Band-Kennsatz
- Prüfung von Schutzattributen bei Banddateien

6.6.1 Löschen, Ändern und Anzeigen von Datei-Attributen

Nur der Datei-Eigentümer (siehe Definition auf "Festlegung von Schutzmerkmalen") kann eine Datei löschen (Makro ERASE, Kommando DELETE-FILE) oder ihre Attribute ändern (Makro CATAL, Kommando MODIFY-FILE-ATTRIBUTES).
Ist für eine Benutzerkennung eine Miteigentümerschaft vergeben (siehe "Festlegung einer Miteigentümerschaft (Co-Owner)"), so haben die Miteigentümer die gleichen Rechte beim Löschen, Ändern und Anzeigen von Datei-Attributen einer Datei wie der Datei-Eigentümer.

Die Benutzerkennung mit dem Privileg TSOS hat (mit Einschränkungen, siehe "Einschränkungen der TSOS-Miteigentümerschaft") die gleichen Zugriffsrechte wie die Eigentümer-Benutzerkennung.
Sie kann für jede Datei – auch ohne Angabe eines vergebenen Kennworts – die Dateiattribute ändern und anzeigen lassen, wenn ihre Miteigentümerschaft nicht eingeschränkt wurde. Für Banddateien gilt das nur, soweit es technisch möglich ist (z.B. keine Änderung der Kennsätze, sofern nicht ein entsprechendes Kopierprogramm benutzt wird).

Die Benutzerkennung mit dem Privileg TSOS kann ein vergebenes Kennwort ermitteln. Bei Einsatz der Kennwortverschlüsselung wird jedoch der verschlüsselte Wert angezeigt, der nicht als Original-Kennwort verwendbar ist.

Eine Datei kann aber nur dann gelöscht werden, wenn Schreibzugriff besteht (ACCESS=WRITE bzw. entsprechende Attribute bei BACL oder GUARDS).
Mit dem Makro ERASE, Operand IGNORE=ACCESS bzw. mit dem Kommando DELETE-FILE, Operand IGNORE=*ACCESS kann ein nicht bestehender Schreibzugriff ignoriert und die Datei gelöscht werden, ohne die Schutzattribute vorher entsprechend zu ändern.

6.6.2 Zugriff zu kennwortgeschützten Dateien

Wenn ein Auftrag auf eine Datei zugreifen will, die mit Kennwörtern geschützt ist, muss das für den Zugriff erforderliche Kennwort angegeben werden, und zwar mit dem Kommando ADD-PASSWORD: die so übergebenen Kennwörter werden in der Kennworttabelle des Auftrags hinterlegt, brauchen also nicht bei jedem Zugriff erneut eingegeben zu werden.

Kennwörter haben nachstehende Rangfolge:

- Schreibkennwort
- Lesekennwort
- Ausführungskennwort

Die Tabelle auf "Zugriff zu kennwortgeschützten Dateien" zeigt, welche Kennwörter jeweils vor einem Zugriff anzugeben sind.

Solange das für den Datei- oder Katalogzugriff benötigte Kennwort in der Kennworttabelle des Auftrags enthalten ist, braucht es vor erneutem Zugriff auf diese Datei oder beim Zugriff auf andere mit diesem Kennwort geschützte Dateien nicht angegeben zu werden. Soll der volle Kennwortschutz wiederhergestellt werden, kann mit dem Kommando REMOVE-PASSWORD ein Kennwort aus der Kennworttabelle oder die gesamte Kennworttabelle gelöscht werden.

Wird nicht explizit mit dem Kommando REMOVE-PASSWORD ein Kennwort aus der Kennworttabelle oder die gesamte Tabelle gelöscht, geschieht dies implizit bei Auftragsende, da die Kennworttabelle auftragsbezogen ist.

Die folgende Tabelle zeigt die möglichen Kombinationen des Kennwortschutzes:

Kennwortschutz	Kennwortangabe	Ausführen	Lesen	Schreiben
EXEC-PASSWORD	keine Angabe	--	--	--
	Ausführungskennwort	X	X	X
READ-PASSWORD	keine Angabe	X*)	--	--
	Lesekennwort	X	X	X
WRITE-PASSWORD	keine Angabe	X	X	--
	Schreibkennwort	X	X	X
EXEC-PASSWORD READ-PASSWORD WRITE-PASSWORD	keine Angabe	--	--	--
	Ausführungskennwort	X*)	--	--
	Lesekennwort	X	X	--
	Schreibkennwort	X	X	X
EXEC-PASSWORD READ-PASSWORD	keine Angabe	--	--	--
	Ausführungskennwort	X*)	--	--
	Lesekennwort	X	X	X

EXEC-PASSWORD WRITE-PASSWORD	keine Angabe	--	--	--
	Ausführungskennwort	X	X	--
	Schreibkennwort	X	X	X
READ-PASSWORD WRITE-PASSWORD	keine Angabe	X*)	--	--
	Lesekennwort	X	X	--
	Schreibkennwort	X	X	X

Tabelle 24: Kennwortangaben, abhängig vom gewünschten Zugriff

X Zugriff möglich

-- Zugriff nicht möglich

*) Der Programmcode ist vor Zugriffen durch Dumps geschützt.

Kennwortgeschützte Dateien löschen

Dateien, die mit einem Kennwort geschützt sind, können erst dann gelöscht werden, wenn der Schreibzugriff durch Kennwortangabe ermöglicht ist (Ausnahme: das FREE-FOR-DELETION-Datum ist abgelaufen, siehe "Festlegung eines Löschzeitpunktes"). Dabei kann das Kennwort mit dem Kommando ADD-PASSWORD in die Kennworttabelle des Auftrags eingetragen oder im Makro ERASE bzw. im Kommando DELETE-FILE angegeben werden.

Sowohl der Makro ERASE als auch das Kommando DELETE-FILE bieten mit dem Operanden PASSWD (Makro) bzw. PASSWORDS-TO-IGNORE die Möglichkeit, ein Kennwort einzugeben, das nur für den aktuellen Makro- oder Kommandoaufruf gilt und nicht in die Kennworttabelle des Auftrags aufgenommen wird. Auf diese Weise bleibt der volle Zugriffsschutz für alle anderen Dateien, die mit diesem Kennwort geschützt wurden und vom Löschen nicht betroffen sind, erhalten.

6.6.3 Zugriff auf eine mit Schutzfrist geschützte Datei

Eine Schutzfrist kann vergeben bzw. geändert werden mit dem Makro CATAL bzw. dem Kommando MODIFY-FILE-ATTRIBUTES (jeweils Eintrag im Dateikatalog) oder mit dem Makro FILE bzw. dem Kommando ADD-FILE-LINK (jeweils Eintrag in der TFT).

Während der Schutzdauer sind auf die Datei keine schreibenden Zugriffe erlaubt. Nur der Eigentümer der Datei kann die Schutzfrist ändern. Mit IGNORE=EXDATE im Makro ERASE bzw. mit IGNORE-PROTECTION=*EXPIRATION-DATE im Kommando DELETE-FILE kann eine Datei mit Schutzfrist gelöscht werden, ohne vorher die Schutzfrist zurückzusetzen.

Ist das FREE-FOR-DELETION-Datum abgelaufen, kann die Datei auch bei noch nicht abgelaufener Schutzfrist gelöscht werden.

6.6.4 Schutz von Mehrdatei-Bändern gegen implizites Löschen

Bei zugelassenem Schreibzugriff auf eine Banddatei werden alle auf diesem Magnetband folgenden Banddateien unabhängig von ihren Schutzattributen implizit gelöscht. Die Katalogeinträge bleiben zwar erhalten, auf die Daten kann jedoch nicht mehr zugegriffen werden.

Bei Ausführung des Makros FILE mit dem Operanden SECLEV=(...,OPR) bzw. des Kommandos ADD-FILE-LINK mit dem Operanden OVERWRITE-PROTECTION=*YES wird bei Schreibzugriff überprüft, welche Schutzattribute für die neue Datei vergeben werden. Wenn die neue Datei nicht als erste auf dem Magnetband abgespeichert wird, werden ihre Schutzattribute mit denen der unmittelbar davor gespeicherten Datei verglichen.

Das Beschreiben des Bandes wird abgelehnt, wenn

- für die aktuelle Datei der Zugriffstyp ACCESS=READ vorgegeben wurde, die unmittelbar davor gespeicherte Datei den Zugriffstyp ACCESS=WRITE besitzt
- die Schutzfrist der aktuellen Datei höher ist, als die der unmittelbar davor gespeicherten Datei

Infolgedessen ist der Schutz der ersten Datei (bzw. der jeweiligen Vorgänger-Datei) auf dem Band maßgeblich für den Schutz der nachfolgenden Dateien gegen Überschreiben.

6.6.5 Übernahme von Schutzattributen in den Band-Kennsatz

Die Bandkennsätze werden bei der Erstellung einer Banddatei erzeugt und können nicht nachträglich geändert werden. Mit dem Makro FILE und dem Operanden LABEL bzw. mit dem Kommando ADD-FILE-LINK und dem Operanden LABEL-TYPE wird festgelegt, welche Dateiattribute in den Bandkennsatz übernommen werden. Mit LABEL=(STD,3) im Makro bzw. mit LABEL-TYPE=*STD(DIN-REV-NUM=3) im Kommando wird der maximal mögliche Schutz erreicht. Die Einstellung LABEL=NO bzw. LABEL-TYPE=*NO ist grundsätzlich zu vermeiden.

6.6.6 Prüfung von Schutzattributen bei Banddateien

Bei der Erstellung einer Banddatei werden die Attribute im Dateikatalog hinterlegt. Wird der Katalogeintrag gelöscht, stehen nur noch die Einträge in den Bandkennsätzen zur Verfügung.

Bei Lesezugriffen und Schreibzugriffen auf Banddateien werden die Einträge im Dateikatalog mit den Einträgen in den Bandkennsätzen verglichen. In Abhängigkeit von der Berechtigung einer Benutzerkennung und den Benutzerangaben für den Zugriff auf die Banddatei (Kommando ADD-FILE-LINK, Operanden LABEL-TYPE bzw. BYPASS-LABEL-CHECK) können die zu Grunde liegenden Sicherheitsmaßnahmen umgangen werden.

6.7 Datenschutz durch „Datenzerstörung" (DESTROY-Option)

Sowohl für Platten als auch für Bänder kann der Benutzer veranlassen, dass nicht mehr benötigte Daten durch Überschreiben mit binär Null „zerstört“ werden. Damit sind diese Daten im BS2000 nicht mehr lesbar, auch nicht mit Diagnoseprogrammen und spezieller Privilegierung.

> **i** Unabhängig von Nutzung dieser Option gilt in **jedem** Fall: Bei Austausch eines Datenträgers (elektronisches, magnetisches und optisches Speichermedium wie Festplatte, Speicherbauteil eines Plattenspeichersystems, Band, etc.), auf dem sensible Daten gespeichert sind oder gespeichert waren, muss nach Löschen der Daten im BS2000 auch der Datenträger selbst zerstört werden. Nur so ist endgültig sichergestellt, dass Daten von diesem Datenträger nicht wieder hergestellt werden können.

Mit dem Makro CATAL bzw. den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES kann im Katalogeintrag Folgendes festgelegt werden:

- Für Plattendateien:
  Speicherplatz, der der Datei zugewiesen ist, wird bei seiner Freigabe explizit gelöscht.
- Für Banddateien:
  Bei Bandwechsel oder nach dem Schließen der Ausgabedatei werden noch folgende Restdaten auf dem Band gelöscht.

Plattendateien

Für Plattendateien kann „Datenzerstörung“ bei Speicherplatzfreigabe definiert werden. Eine solche Option kann entweder im Dateikatalog eingetragen oder explizit beim Löschen von Dateien angegeben werden.

im Katalogeintrag:	CATAL ...,DESTROY=*YES /CREATE-FILE ...,DESTROY-BY-DELETE=*YES /MODIFY-FILE-ATTRIBUTES ...,DESTROY-BY-DELETE=*YES
beim Löschen:	ERASE ...,DESTROY /DELETE-FILE ...,OPTION=*DESTROY-ALL

Banddateien

Für Banddateien kann im Makro FILE (Operand DESTOC=YES) bzw. im Kommando ADD-FILE-LINK (Operand DESTROY-OLD-CONTENTS=*YES) festgelegt werden, dass nach Schließen der Ausgabedatei (CLOSE) oder bei Bandwechsel weitere auf dem Band noch folgende (alte) Daten überschrieben, d.h. physikalisch gelöscht werden.

Automatische „Datenzerstörung“ bei Rekonstruktion

Automatische „Datenzerstörung“ führt das DVS bei der Rekonstruktion von Dateien durch, die z.B. infolge Systemabbruchs zerstört sind: Wenn die zu rekonstruierende Datei auf Privatplatte steht oder nicht unter der Benutzerkennung des Auftrags katalogisiert ist, werden die vom DVS zur Rekonstruktion angelegten Hilfsdateien mit Datenzerstörung gelöscht (siehe "Rekonstruktion von Dateien mit fremder Benutzerkennung").

6.8 Dateiverschlüsselung

Outsourcing, Serverkonsolidierung und Storage-Virtualisierung haben zu der Anforderung geführt, den Inhalt von Dateien auch auf Sicherungen und auch gegenüber Personen mit Privileg TSOS zu schützen. Diese Anforderung wird durch eine ins Dateisystem integrierte Dateiverschlüsselung realisiert.

6.8.1 Konzept der Dateiverschlüsselung

Die Verschlüsselung erfolgt mit einem der symmetrischen Standardverfahren AES oder DES (das Verschlüsselungsverfahren wird als Systemoption im Systemparameter FILECRYP festgelegt) und einem einzugebenden Crypto-Kennwort. Um auf den unverschlüsselten Inhalt einer verschlüsselten Datei zugreifen zu können, muss dieses Crypto-Kennwort tasklokal eingegeben werden.

Die Verschlüsselung einer Datei wird bei den Dateiattributen als Schutzattribut geführt, betrifft aber technisch auch das Dateiformat.

Der Systemparameter FILECRYP kann jederzeit geändert werden, etwa von AES auf DES. Der Systemparameter wirkt sich (nur) aus auf die Wahl des Verfahrens bei der Umwandlung in eine verschlüsselte Datei (ENCRYPT-FILE). Bereits verschlüsselte Dateien führen ihre Verschlüsselungsmerkmale (auch das benutzte Verfahren) im Katalogeintrag und bleiben von der Änderung des Systemparameters unberührt.

Verschlüsselte Dateien werden seitenweise verschlüsselt auf den Pubset-Platten geführt. Dabei sind alle Seiten einer Datei gleichartig verschlüsselt. Aus der Prüfung des Crypto-Kennworts beim Öffnen einer verschlüsselten Datei wird der Schlüssel (Key) zum Ver- bzw. Entschlüsseln für die Lese- oder Schreibzugriffe auf die Datei abgeleitet.

Ver- bzw. Entschlüsseln erfolgt über das Subsystem Crypt.

Verschiedene Dateien können gleiche oder unterschiedliche Crypto-Kennwörter haben. Dadurch werden gleiche oder unterschiedliche Keys beim Verschlüsseln verwendet.
Bei der Umwandlung in eine verschlüsselte Datei (ENCRYPT-FILE) kann man das Crypto-Kennwort auch von einer schon verschlüsselten Referenzdatei übernehmen. Damit wird die Vergabe von gleichen Crypto-Kennwörtern für zusammengehörende Dateien unterstützt. Mit dem Systemparameter FREFCRYP kann man die Einführung neuer Crypto-Kennwörter beschränken auf Dateien einer bestimmten Benutzerkennung. Dateien außerhalb dieser Benutzerkennung können dann nur ein von einer Referenzdatei übernommenes Crypto-Kennwort erhalten.
Der Katalogeintrag einer verschlüsselten Datei enthält den Verschlüsselungsindikator, das Verschlüsselungsverfahren und ein Prüfmuster zur Prüfung des Crypto-Kennworts.
Das Crypto-Kennwort und der Key werden nicht im System gespeichert.

Beim Einsatz von DAB (Disk Access Buffer) werden durch die Pufferung die Ein-/Ausgaben auf Platten reduziert. Bei Zugriffen auf verschlüsselte Dateien mit DAB-Pufferung wird neben den Ein-/Ausgaben auf Platte auch die Ver- bzw. Entschlüsselung gespart.

Homogener Transfer und Integration in bestehende Anwendungen

Beim **homogenen Transfer** einer verschlüsselten Datei wird der verschlüsselte Inhalt eins-zu-eins in eine Empfangsdatei übertragen, die dieselben Verschlüsselungsattribute hat wie die Sendedatei.

Dieser homogene Transfer wird benutzt bei:

- homogenem COPY-FILE
- Sichern und Restaurieren (SAVE/RESTORE) mit HSMS/ARCHIVE
- Verdrängen und Zurückholen (MIGRATION/RECALL) mit HSMS
- Verlagern von Dateien innerhalb eines SM-Pubsets auf einen anderen Volume-Set
- File-Transfer (nur homogen; für den inhomogenen File-Transfer siehe Einschränkung auf "Voraussetzungen und Einschränkungen für Dateiverschlüsselung")

Beim homogenen Transfer findet also keine Entschlüsselung statt und somit wird auch kein Key und kein Crypto-Kennwort benötigt. Die Zieldatei wird verschlüsselt angelegt, wobei nicht mehr Zeit erforderlich ist als für die entsprechende Operation für eine unverschlüsselte Datei.

Der Systembetreuer kann einerseits mit verschlüsselten Dateien wie gewohnt arbeiten, da bei den genannten Transfers kein Crypto-Kennwort benötigt wird, er hat jedoch andererseits keinen Zugriff auf den entschlüsselten Inhalt von verschlüsselten Dateien.

Für die einfache Einführung verschlüsselter Dateien in vorhandene Anwendungen werden sicherheitskritische Dateien einmalig in verschlüsselte Dateien umgewandelt. Dabei muss ein Crypto-Kennwort festgelegt werden. Wie bei den Dateikennwörtern kann die Vorgabe des Crypto-Kennworts außerhalb der Anwendung erfolgen, bevor die Anwendung dann in der gleichen Task gestartet wird. Die Anwendung liest und schreibt dann bei Dateizugriffen automatisch den entschlüsselten Inhalt der Datei. Auf den Platten ist und bleibt der Inhalt der Datei verschlüsselt. Auf diese Weise brauchen Anwendungen bei Nutzung von verschlüsselten Dateien nicht geändert werden.

6.8.2 Voraussetzungen und Einschränkungen für Dateiverschlüsselung

BS2000 unterstützt die Verwendung verschlüsselter Dateien über das Subsystem CRYPT (siehe Handbuch „CRYPT“ [24]). CRYPT kann im gleichen System für andere Zwecke mitgenutzt werden (z.B. für die Verschlüsselung bei File-Transfer oder IPSec).

Wenn eine verschlüsselte Datei auf einem Shared-Pubset liegt, müssen auf allen Systemen, von denen aus auf die Datei ein Zugriff mit Entschlüsseln erfolgen soll, die Voraussetzungen für den Betrieb verschlüsselter Dateien erfüllt sein.

Vorhandene Anwendungen, die mit verschlüsselten Dateien arbeiten sollen, müssen in der Regel nicht geändert werden. Änderungen in Anwendungen sind nur dann notwendig, wenn neue, verschlüsselte Dateien von der Anwendung angelegt werden sollen.

Eingeschränkter Zugriff

Pubsets mit verschlüsselten Dateien können auch importiert werden, wenn dem System die Voraussetzungen für Dateiverschlüsselung fehlen (z.B. Subsystem CRYPT ist nicht verfügbar).

In diesem Fall sind Zugriffe auf diese Dateien mit Entschlüsselung des Inhalts nicht möglich. Eine Änderung von Dateiattributen ist ebenfalls nicht möglich. Die verschlüsselten Dateien können nur angezeigt (SHOW-FILE-ATTRIBUTES) und gelöscht (DELETE-FILE) werden. Der homogene Transfer dieser Dateien (siehe "Konzept der Dateiverschlüsselung") ist möglich.

Umfang und Einsatz verschlüsselter Dateien

Es können nur Plattendateien auf Pubsets verschlüsselt werden.

Verschlüsselte Dateien können für alle Zugriffsarten (Ausnahme EAM und BTAM), alle Dateitypen, alle Dateiformate und alle Pubset-Typen eingesetzt werden.

Für die Wiederherstellung von ISAM-Dateien (VERIFY) ist die Vorgabe des Crypto-Kennworts nötig, ebenso für die Konvertierung mit PAMCONV.

Verträglichkeit mit anderen Schutzmechanismen

Wenn eine mit Lese- und/oder Ausführungskennwort geschützte Datei in eine verschlüsselte Datei umgewandelt wird, so verliert sie den Schutz durch Lese- und/oder Ausführungskennwort.

Die Dateiverschlüsselung ist mit dem Schreibkennwort (WRITE-PASSWORD) und mit allen anderen Schutzmechanismen kombinierbar. Zur gewohnten Nutzung dieser Schutzmechanismen kommt das Crypto-Kennwort hinzu.

Einschränkungen und Besonderheiten

- Durch die Bindung an das Subsystem CRYPT ist erst ab „System Ready“ der Zugriff auf verschlüsselte Dateien möglich.
- Jobvariablen, Banddateien und Dateien auf Net-Storage oder Privatplatten können nicht verschlüsselt werden.
- Dateien der Benutzerkennung TSOS auf dem Home-Pubset können nicht verschlüsselt werden.
- Eine verschlüsselte Datei muss erst entschlüsselt werden, bevor sie gedruckt werden kann (siehe auch „Ausdrucken einer verschlüsselten Datei“ (Einsatz-Szenarien)).
- Für verschlüsselte PLAM-Bibliotheken wird beim HSMS-Backup die Option SAVE-PLAM-INFO nicht ausgeführt.

- Die Konvertierung vom K-Format in ein NK-Format beim Restaurieren oder Importieren einer K-Datei von einem Sicherungsdatenträger auf eine NK-Platte ist nicht möglich, wenn die Datei verschlüsselt ist.
- Dateien mit einem gültigen Last Byte Pointer werden nicht verschlüsselt.

6.8.3 Benutzerschnittstellen

Für Dateiverschlüsselung stehen folgende DVS-Kommandos und Makros zur Verfügung:

Kommando	Bedeutung
ADD-CRYPTO-PASSWORD	hinterlegt das Crypto-Kennwort zur Entschlüsselung von verschlüsselten Dateien in der Crypto-Kennworttabelle der Task
DECRYPT-FILE	wandelt verschlüsselte Datei in unverschlüsselte Datei um
ENCRYPT-FILE	wandelt unverschlüsselte Datei in verschlüsselte Datei um
REMOVE-CRYPTO-PASSWORD	entfernt das Crypto-Kennwort aus der Crypto-Kennworttabelle der laufenden Task
Makro	**Bedeutung**
DECFILE	wandelt verschlüsselte Datei in unverschlüsselte Datei um
ENCFILE	wandelt unverschlüsselte Datei in verschlüsselte Datei um

Verschlüsselung

Mit dem Makro ENCFILE oder dem Kommando ENCRYPT-FILE wird eine unverschlüsselte Datei in eine verschlüsselte Datei umgewandelt. Wird eine bereits verschlüsselte Referenzdatei an Stelle des Crypto-Kennworts angegeben, dann erhält die Datei die gleichen Verschlüsselungsattribute wie die Referenzdatei, insbesondere das gleiche Crypto-Kennwort. Für die Verschlüsselung muss die Datei nur katalogisiert sein.

Bei der Umwandlung einer unverschlüsselten in eine verschlüsselte Datei werden Lese- und/oder Ausführungskennwort (READ-/EXEC-PASSWORD) gelöscht.

Das im Systemparameter festgelegt Verschlüsselungsverfahren (AES oder DES) wird bei der Umwandlung in eine verschlüsselte Datei verwendet und dann an die Datei gebunden.

Die Verschlüsselung ist für eigene Dateien oder für Dateien in Miteigentümerschaft möglich (wie die Berechtigung zum Anlegen einer Datei) und kann nur lokal und nicht über RFA erfolgen.

> **!** **Achtung!**
>
> Ist das Crypto-Kennwort nicht mehr bekannt, ist keine Entschlüsselung des Dateiinhalts möglich! Deshalb sollte das Crypto-Kennwort an einem sicheren Ort hinterlegt werden (siehe auch „Administration der Verschlüsselung“ (Einsatz-Szenarien)).

Zugriff auf verschlüsselte Dateien

Das Kommando ADD-CRYPTO-PASSWORD hinterlegt das Crypto-Kennwort in der Crypto-Kennworttabelle der Task. Nur mit dem korrekten Crypto-Kennwort ist das Öffnen einer Datei zum Zugriff auf den unverschlüsselten Inhalt erlaubt.

Beim einfachen Verlagern einer verschlüsselten Datei ist kein Crypto-Kennwort notwendig.

Die Crypto-Kennwörter werden nur in festverschlüsselter Form in der Crypto-Kennworttabelle hinterlegt, und zwar unabhängig vom Systemparameter zum Verschlüsseln von Dateikennwörtern.

Enthält eine Prozedur ein Crypto-Kennwort (z.B. in einem Kommando ADD-CRYPTO-PASSWORD) und soll deswegen für Dritte – insbesondere für den Ausführenden – unleserlich gemacht werden, wird der Einsatz von SDF-P-Compilierung mit automatischer Festverschlüsselung empfohlen (siehe „Verarbeiten von verschlüsselten Dateien in Prozeduren“ (Einsatz-Szenarien)).
Zum Einsatz von SDF-P siehe Handbuch „SDF-P“ [17].

Mit dem Kommando REMOVE-CRYPTO-PASSWORD wird ein Crypto-Kennwort aus der Crypto-Kennworttabelle der laufenden Task entfernt. Das Entschlüsseln von Dateien, die mit diesem Crypto-Kennwort verschlüsselt sind, ist dann nicht möglich.

Die Kommandos ADD-/REMOVE-CRYPTO-PASSWORD werden automatisch an alle RFA-Partnerprozesse weitergeleitet.

Entschlüsselung

Mit dem Makro DECFILE oder dem Kommando DECRYPT-FILE wird eine verschlüsselte Datei wieder zu einer unverschlüsselten Datei. Vor dem Aufruf muss das Crypto-Kennwort vorgegeben werden.

Die Entschlüsselung ist für eigene Dateien oder für Dateien in Miteigentümerschaft möglich (wie die Berechtigung zum Anlegen einer Datei) und kann nur lokal und nicht über RFA erfolgen.

Weitere Abhängigkeiten

CONCATENATE-DISK-FILES
Für die Verkettung verschlüsselter SAM-Dateien ist die Vorgabe der Crypto-Kennwörter notwendig (mit ADD-CRYPTO-PASSWORD).

COPY-FILE
Die Verschlüsselungsmerkmale werden – soweit möglich – von der Ausgangsdatei in die Zieldatei übernommen. Der Inhalt der Datei wird eins-zu-eins kopiert. Die Vorgabe des Crypto-Kennworts ist nicht notwendig.

Wenn die Verschlüsselungsmerkmale nicht übernommen werden können, ist die Vorgabe des Crypto-Kennworts notwendig (ADD-CRYPTO-PASSWORD). Das ist z.B. dann der Fall, wenn die Zieldatei nicht verschlüsselt werden kann oder wenn das Ziel eine Dateigeneration mit den einheitlichen Dateischutzattributen der Dateigenerationsgruppe ist.

DELETE-FILE
Es wird eine Dateiselektion abhängig von der Dateiverschlüsselung angeboten, also nach den Werten vom Dateimerkmal ENCRYPTION.

MODIFY-FILE-ATTRIBUTES
Falls eine verschlüsselte Datei einen Katalogeintrag, jedoch noch keinen Speicherplatz hat, darf ihr kein Speicherplatz auf Privatplatte und kein Bandtyp zugewiesen werden. Verschlüsselungsmerkmale können nicht geändert werden. Änderungen des Lese- und Ausführungskennworts werden bei verschlüsselten Dateien ignoriert.

REPAIR-DISK-FILES
Um verschlüsselte ISAM-Dateien reparieren zu können, ist die Vorgabe des zugehörigen Crypto-Kennworts notwendig (ADD-CRYPTO-PASSWORD). Die Dateikopie erhält – soweit möglich – die gleichen Verschlüsselungsmerkmale.

SHOW-FILE
Der Inhalt einer verschlüsselten Datei wird entschlüsselt angezeigt. Hierbei ist die Vorgabe des Crypto-Kennworts notwendig (ADD-CRYPTO-PASSWORD).

SHOW-FILE-ATTRIBUTES

Das Dateimerkmal ENCRYPTION im Abschnitt SECURITY zeigt an, ob und mit welchem Verfahren (AES oder DES) eine Datei verschlüsselt ist. Eine Dateiselektion wird entsprechend dieser Werte angeboten.

6.8.4 Einsatz-Szenarien

Die Datei AUSLANDSKONTEN wird als „sicherheitsrelevant“ eingestuft und soll deshalb verschlüsselt geführt werden. Sie wird einmalig in eine verschlüsselte Datei umgewandelt:

```
/ENCRYPT-FILE FILE-NAME=AUSLANDSKONTEN, -
              CRYPTO-PASSWORD='KROKODIL',CONFIRM-PASSWORD='KROKODIL'
```

Danach kann man kontrollieren, ob und mit welcher Verschlüsselungsmethode die Datei verschlüsselt wurde:

```
/SHOW-FILE-ATTRIBUTES AUSLANDSKONTEN,SECURITY=*YES
0000000066 :X:$U234.AUSLANDSKONTEN
------------------------------ SECURITY     ------------------------------
   READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
   USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
   AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2015-02-08
   DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
   SP-REL-LOCK= NO          ENCRYPTION = AES
 :X:    PUBLIC:      1 FILE  RES=        66 FRE=       50 REL=       18 PAGES
```

Die folgenden Einsatz-Szenarien zeigen die verschiedenen Anforderungen an Anwendungen oder Benutzer, die sich beim Arbeiten mit verschlüsselten Dateien ergeben.
Einige der Anwendungen müssen angepasst werden, bei anderen genügt die Vorgabe des Crypto-Kennworts zum Arbeiten mit verschlüsselten Dateien.

Folgende Einsatz-Szenarien werden dargestellt:

- Anwendung schreibt in verschlüsselter Datei
- Anwendung liest in verschlüsselter Datei und erzeugt verschlüsselte Ausgabedatei
- Verarbeiten von verschlüsselten Dateien in Prozeduren
- Weiterreichen der verschlüsselten Datei an Subtasks
- Ausdrucken einer verschlüsselten Datei
- Administration der Verschlüsselung

Anwendung schreibt in verschlüsselter Datei

Die Anwendung UPDATE.AUSL-KONTEN trägt die aktuellen Kontostände in der Datei AUSLANDSKONTEN nach. Die Anwendung kann unverändert bleiben, auch wenn die Datei AUSLANDSKONTEN jetzt verschlüsselt ist. Vor dem Aufruf der Anwendung muss lediglich das Crypto-Kennwort vorgegeben werden:

```
/ADD-CRYPTO-PASSWORD 'KROKODIL'
/START-EXEC-PROGRAM UPDATE.AUSL-KONTEN
/REMOVE-CRYPTO-PASSWORD 'KROKODIL'
```

Anwendung liest in verschlüsselter Datei und erzeugt verschlüsselte Ausgabedatei

Die Anwendung KONTSORT sortiert die Konten nach den aktuellen Kontoständen und erzeugt dafür eine Ausgabedatei. Ist die Eingabedatei verschlüsselt (wie z.B. die Datei AUSLANDSKONTEN), soll die Ausgabedatei in gleicher Weise verschlüsselt werden.
Dazu muss KONTSORT geändert werden.

Nach dem Anlegen und vor dem Öffnen der Ausgabedatei ist das Umwandeln in eine verschlüsselte Datei einzufügen:

```
FILE AUSGABEDATEI
ENCFILE PATHNAM='AUSGABEDATEI',REFFILE='EINGABEDATEI'
OPEN AUSGABEDATEI
```

Wenn die Eingabedatei nicht verschlüsselt ist, wird die Ausführung des Makros ENCFILE abgewiesen und die Ausgabedatei wird nicht verschlüsselt.
Wenn die Eingabedatei verschlüsselt ist, erhält die verschlüsselte Ausgabedatei das gleiche Crypto-Kennwort wie die Eingabedatei. Dieses muss beim Aufruf der Anwendung KONTSORT vorgegeben sein (wie in „Anwendung schreibt in verschlüsselter Datei“), damit KONTSORT die Eingabedatei öffnen kann.

Verarbeiten von verschlüsselten Dateien in Prozeduren

Das Programm UPDATE.AUSL-KONTEN (siehe „Anwendung schreibt in verschlüsselterDatei“) soll in der S-Prozedur UPDATE.PROC aufgerufen werden, die täglich vom Systembetreuer zum Ablauf gebracht wird. In die S-Prozedur ist zwar die Anweisung ADD-CRYPTO-PASSWORD='KROKODIL' eingebracht worden, der Systembetreuer selber soll aber keine Kenntnis vom Crypto-Kennwort erhalten.

Vorgehensweise: Die S-Prozedur UPDATE.PROC wird mit COMPILE-PROCEDURE konvertiert (hierfür wird SDF-P benötigt). Ergebnis ist eine kompilierte Prozedurdatei, in der das Crypto-Kennwort nicht mehr lesbar ist. Dem Systembetreuer wird nur diese Datei zugänglich gemacht. Er kann sie ausführen, aber nicht lesen.

Weiterreichen der verschlüsselten Datei an Subtasks

Das Programm STEUER.AUSL-KONTEN wertet jedes Konto der Datei AUSLANDSKONTEN aus und erstellt dafür jeweils Steuerbescheinigungen. Das Programm arbeitet asynchron in mehreren Subtasks und öffnet dort die verschlüsselte Datei AUSLANDSKONTEN. Das Vorgeben des Crypto-Kennworts über die taskspezifische Crypto-Kennworttabelle ist dafür ungeeignet, weil die Berechtigung nicht von der aufrufenden Task, sondern in den Subtasks gebraucht wird. Also muss das Programm geändert werden.

Es wird ein Programm-Parameter eingeführt, über den das Crypto-Kennwort angegeben werden kann. Der Programmaufruf sieht dann wie folgt aus:

```
/START-EXEC-PROGRAM STEUER.AUSL-KONTEN

*Crypto-Kennwort: 'krokodil'
```

Dann muss das Programm das angegebene Crypto-Kennwort an die Subtasks weiterreichen, welche die Datei AUSLANDSKONTEN öffnen.
In der Subtask muss das Programm vor dem Öffnen der Datei AUSLANDSKONTEN das Crypto-Kennwort über den P1-FCB vorgeben (zur Beschreibung des Makros FCB siehe Handbuch „DVS-Makros“ [1]).

Ausdrucken einer verschlüsselten Datei

Beim Ausdrucken einer verschlüsselten Datei (per Hand oder aus einem Programm heraus) sind – besonders bei zentralem Druck – Sicherheitsaspekte zu berücksichtigen. Diesen Sicherheitsaspekten kann man z.B. durch die Einschränkung auf Ausdrucke nur per Hand und nur auf einen lokalen Drucker im gleichen Zimmer genügen.

Eine verschlüsselte Datei kann man nicht direkt ausdrucken. Der Benutzer muss vorübergehend eine unverschlüsselte Kopie der auszudruckenden Datei erzeugen und diese Kopie drucken. Nach dem Drucken der Kopie sollte ihr Inhalt mit binär null überschrieben werden.

```
/ADD-CRYPTO-PASSWORD 'KROKODIL'
/COPY-FILE AUSLANDSKONTEN,TEMP-DRUCK
/DECRYPT-FILE TEMP-DRUCK
/REMOVE-CRYPTO-PASSWORD 'KROKODIL'
/PRINT-DOCUMENT TEMP-DRUCK,DELETE-AFTER-PRINT=*DESTROY
```

Anmerkung

Wird die Kopie TEMP-DRUCK vor dem COPY-FILE als Privatplattendatei angelegt, so wird schon bei COPY-FILE der Dateiinhalt entschlüsselt (weil eine Privatplattendatei nicht verschlüsselt sein darf), sodass DECRYPT-FILE entfällt.

Administration der Verschlüsselung

Die Revisionsabteilung soll die Nutzung von verschlüsselten Dateien überwachen, die Vergabe von Crypto-Kennwörtern kontrollieren und sie sicher gegen Verlust aufbewahren.

1. Die Revisionsabteilung richtet auf den Pubsets, die verschlüsselte Dateien enthalten sollen, eine eigene Kennung MUSTER ein.
   Außerdem setzt sie in den Systemen, in denen Verschlüsselung genutzt werden soll, den Systemparameter FREFCRYP auf den Wert „MUSTER“.
   Nun können außerhalb der Kennung MUSTER verschlüsselte Dateien nur noch eingerichtet werden mit Bezug auf schon verschlüsselte Dateien und nicht mehr mit frei wählbaren Crypto-Kennwörtern.
2. Die Abteilung für Auslandskonten beantragt bei der Revisionsabteilung die Nutzung von verschlüsselten Dateien.
3. Die Revisionsabteilung genehmigt das und richtet eine mehrbenutzbare Datei mit dem Namen $MUSTER.REFERENZ.AUSLANDSKONTEN ein. Der Dateiinhalt spielt keine Rolle.
4. Der Vertreter der Auslandskonten wandelt diese Datei in eine verschlüsselte Datei um und definiert ein Crypto-Kennwort nach seiner Wahl.
5. Ein Hardcopy von diesem ENCRYPT-FILE-Aufruf (mit dem definierten Crypto-Kennwort) wird versiegelt und im Tresor der Revisionsabteilung hinterlegt.
6. Nun kann die Abteilung für Auslandskonten auf ihren Kennungen nach Bedarf Dateien mit Bezug auf diese Referenzdatei verschlüsseln.
   Ein Bezug auf andere Referenzdateien ist der Abteilung nicht möglich, wenn sie das zugehörige Crypto-Kennwort nicht kennt.
7. Als Hauskonvention könnten in der Referenzdatei z.B. die Mitarbeiter geführt werden, welche das zugehörige Crypto-Kennwort erfahren haben.

6.8.5 Performance, Lastausgleich, Ausfallsituationen

Die Ver- bzw. Entschlüsselung verlangsamt die Dateizugriffe, als ob der Plattenzugriff insgesamt langsamer wäre. Der Einsatz von DAB ist vorteilhaft, weil durch die eingesparten Plattenzugriffe jeweils auch das Ver- bzw. Entschlüsseln entfällt.

7 Datensicherheit

7.1 Schutz bei Parallelzugriff

Das DVS bietet außer dem Schutz einer Datei durch Katalogmerkmale zusätzlich Mechanismen, die eine gegenseitige Störung von Aufträgen unterbinden, wenn diese gleichzeitig eine Datei bearbeiten wollen.

- Mehrere Aufträge können gleichzeitig eine Datei lesen (die Schutzmerkmale der Datei werden dabei natürlich berücksichtigt).
- Bei den Zugriffsmethoden ISAM, UPAM, FASTPAM und DIV (siehe Beschreibung der einzelnen Zugriffsmethoden) bietet das DVS die Funktion „Shared-Update-Verarbeitung“. Dabei können mehrere Aufträge gleichzeitig eine Datei lesend und schreibend bearbeiten. Um Störungen zwischen den Aufträgen zu vermeiden, können Teile der Datei von einem Auftrag gesperrt werden.

Außer diesen internen Schutzmechanismen kann der Benutzer selbst seine Daten vor mehrfachen Zugriffen schützen, indem er Datenträger oder Dateien mit dem Kommando SECURE-RESOURCE-ALLOCATION reserviert. Dies gilt besonders für Dateigenerationsgruppen, wenn die Datenkonsistenz zwischen den Generationen erhalten bleiben soll (siehe Abschnitt „Pubset-Sicherung mit Snapsets“).

7.2 Dateien rekonstruieren

Dateien können für spätere Zugriffe unzugänglich werden, wenn die Verarbeitung nicht normal beendet wird, weil z. B. der bearbeitende Auftrag wegen Systemzusammenbruchs abgebrochen oder wegen Speicherplatzsättigung bis zum Ende des Systemlaufs deaktiviert wird.

Mögliche Folgen der abnormalen Beendigung der Dateiverarbeitung:

- Die Dateisperre wird nicht aufgehoben.
- Dateiinhalt und Katalogeintrag stimmen nicht überein (z.B.: Last Page Pointer).
- Index- und Datenteil von ISAM-Dateien sind nicht konsistent.

Mit dem Makro VERIF bzw. mit dem Kommando REPAIR-DISK-FILE können derartige Dateien rekonstruiert werden.

Versucht ein Programm, auf gesperrte oder zerstörte Dateien zuzugreifen, kann über den EXLST-Ausgang OPENC (siehe Makro EXLST) eine Fehlerroutine aktiviert werden, die den VERIF-Makroaufruf enthält.

Mit dem Makro FSTAT (Operand STATE=NOCLOS) bzw. dem Kommando SHOW-FILE-ATTRIBUTES (Operand STATUS=*PAR(CLOSED-OUTPUT=*NO) kann festgestellt werden, ob und welche Dateien geöffnet sind.

7.2.1 Dateisperre aufheben

„Dateisperre aufheben“ heißt: der Eintrag in der File Lock Table (Tabelle der gesperrten Dateien) wird gelöscht. Dafür steht die VERIFY-Funktion (Makro VERIF bzw. Kommando REMOVE-FILE-ALLOCATION-LOCKS) zur Verfügung. Eine Datei kann mit dem Makro bzw. dem Kommando nur dann entsperrt werden, wenn der bearbeitende Auftrag anhaltend inaktiv ist und die Datei durch ein SECURE-RESOURCE-ALLOCATION-Kommando gesperrt wurde oder wenn die Datei eine Banddatei ist (Kommando SECURE-RESOURCE-ALLOCATION).

Nach einem Systemabbruch existieren evtl. Dateisperren für Dateien, die auf einem Shared-Pubset liegen.

Plattendateien, die nicht durch ein SECURE-RESOURCE-ALLOCATION-Kommando gesperrt wurden, können nur von der Systembetreuung unter der Benutzerkennung TSOS entsperrt werden. Hierbei muss sichergestellt sein, dass niemand mit der Datei arbeitet.

Bereits entsperrte Plattendateien werden beim Makro VERIF bzw. beim Kommando REPAIR-DISK-FILE je nach Zugriffsmethode, mit der sie erstellt wurden, unterschiedlich behandelt.

- SAM-Dateien: im Katalogeintrag wird der Last Page Pointer aktualisiert, sodass die Datei auch im EXTEND-Modus wieder eröffnet werden kann. Es können jedoch die Sätze verloren gegangen sein, die sich bei Auftragsabbruch noch im Ein-/Ausgabepuffer befanden und noch nicht zur Datei übertragen wurden (siehe Abschnitt „Grenzen derRekonstruktion von ISAM-Dateien“).
- ISAM-Dateien: alle noch rekonstruierbaren Sätze werden in eine neue ISAM-Datei (Wiederherstellungsdatei) geschrieben, die nicht-interpretierbaren Datenblöcke in eine PAM-Datei (siehe Abschnitt „Grenzen der Rekonstruktion von ISAM-Dateien“).

  Der Benutzer kann festlegen, welche Datei als Wiederherstellungsdatei genutzt werden soll. Liegt diese Datei auf Privatplatte, muss ausreichend Speicherplatz zur Verfügung stehen. Wird im Makro VERIF bzw. im Kommando REPAIR-DISK-FILE eine Wiederherstellungsdatei definiert, wird die beschädigte Datei nicht überschrieben oder gelöscht.

  Wird keine Wiederherstellungsdatei zur Verfügung gestellt, wird die ISAM-Datei in einer Arbeitsdatei auf gemeinschaftlichen Platten rekonstruiert. Die weitere Behandlung von Original- und Arbeitsdatei hängt davon ab, auf welchen Datenträgern die beschädigte Datei gespeichert ist:

  - Steht die beschädigte Datei auf gemeinschaftlichen Platten, wird sie gelöscht (aus Performance-Gründen ohne Datenzerstörung), und die Arbeitsdatei erhält den Namen der Originaldatei.
  - Steht die beschädigte Datei auf privaten Datenträgern, wird sie vom Inhalt der Arbeitsdatei überschrieben. Anschließend wird die Arbeitsdatei gelöscht (wegen des Datenschutzes mit Datenzerstörung).
- PAM-Dateien: das DVS führt eine privilegierte Schließoperation durch und aktualisiert den Last Page Pointer.

  Für Dateien mit BLOCK-CONTROL-INFO=NO wird der Last Page Pointer auf die letzte belegte Seite gesetzt. Für Dateien mit BLOCK-CONTROL-INFO=WITHIN-DATA-BLOCK wird die Datei vom Ende aus rückwärts gelesen, und der Last Page Pointer auf die letzte Seite gesetzt, die logisch zur Datei gehört. Dies kann bei großen Dateien sehr zeitaufwändig sein.

  Der Last Byte Pointer wird auf Null gesetzt.

7.2.2 Grenzen der Rekonstruktion von ISAM-Dateien

Die Pufferung von Datensätzen von SAM- und ISAM-Dateien im Arbeitsspeicher kann zur Folge haben, dass unmittelbar vor Abbruch der Dateibearbeitung durchgeführte Änderungen verloren gehen. Bei ISAM-Dateien kann diesem Problem mit der WROUT-Funktion begegnet werden (siehe "WROUT-Funktion"). Zu beachten ist aber der erhöhte Zeitaufwand.

Bei der Wiederherstellung von ISAM-Dateien wird von mehrfach vorhandenen Sätzen (d.h. Sätzen mit gleichen Schlüsseln und gleichen Daten) jeweils nur einer in die Wiederherstellungsdatei übernommen. Diese Einschränkung ist notwendig, um ISAM-Dateien wiederherstellen zu können, deren Bearbeitung während eines Blocksplittings abgebrochen wurde.

Enthält eine beschädigte ISAM-Datei mehrere Sätze mit gleichen Schlüsseln (aber verschiedenen Daten), so kann die Reihenfolge dieser Sätze in der Wiederherstellungsdatei von der ursprünglichen Reihenfolge abweichen.

Wenn die zu rekonstruierende ISAM-Datei mit einem Crypto-Kennwort verschlüsselt war, werden ihre Verschlüsselungsattribute auf die wiederhergestellte Datei übertragen.

Kann ein Datenblock einer ISAM-Datei nicht rekonstruiert werden (z.B. weil interne Zeiger zerstört worden sind), wird er in eine gesonderte PAM-Datei geschrieben. Diese PAM-Datei steht dem Benutzer nach Abschluss der Wiederherstellung zu eigenen Rekonstruktionen zur Verfügung. Sie heißt: „S.dateiname.REPAIR“ („dateiname“ ist der Dateiname der beschädigten Datei).

Bei der Wiederherstellung von ISAM-Dateien wird in den Datenblöcken kein Platz für spätere Erweiterungen frei gehalten.

Der Zeitbedarf für die Wiederherstellung von ISAM-Dateien ist nicht zu vernachlässigen. Es wird insbesondere bei größeren ISAM-Dateien empfohlen, die zu bearbeitenden Dateien in einen DAB-Cache zu legen. Die Datei SYSPRC.BS2CP.vvv.TEMPLATE enthält ein Prozedur-Template, das für diesen Zweck benutzt und auf die individuellen Bedürfnisse angepasst werden kann.
Bei Privatplatten kann sich der Zeitbedarf beträchtlich vergrößern, wenn, vom Benutzer keine (bereits mit genügend Speicherplatz versehene) Wiederherstellungsdatei zur Verfügung gestellt wird.

ISAM-Dateien auf verschiedenen Privatplatten für Index- und Datenteil können nur wiederhergestellt werden, wenn für den Katalogeintrag gilt: BUF-LEN=STD/(STD,1).

7.2.3 Rekonstruktion von Dateien mit fremder Benutzerkennung

Soll eine unter fremder Benutzerkennung katalogisierte Datei rekonstruiert werden, muss sie mehrbenutzbar sein. Standardmäßig legt das DVS – falls nicht anders verlangt – eine Hilfsdatei unter der Benutzerkennung des Aufrufers an. Diese Hilfsdatei wird nach Wiederherstellung in die Ursprungsdatei kopiert und mit Datenzerstörung gelöscht. Auch eine evtl. nötige dritte Datei wird unter der Kennung des aufrufenden Auftrags angelegt. Die Hilfsdateien werden auf dem Datenträger eingerichtet, in dem die zu reparierende Datei liegt. Der Aufrufer muss also für den Datenträger der angegebenen Datei zugriffsberechtigt sein.

7.3 Fixpunkt- und Wiederanlauf-Verarbeitung

In Programmen können mit dem Makroaufruf WRCPT Sicherungspunkte gesetzt werden, an denen Programmzustand, Systemumgebung usw. in eine Fixpunktdatei festgeschrieben werden. Für große Banddateien kann über den Makro FILE (Operand CHKPT) bzw. über das Kommando ADD-FILE-LINK (Operand CHECKPOINT-WRITE) automatisches Fixpunktschreiben veranlasst werden.

Mit dem Kommando RESTART-PROGRAM kann bei Programmabbruch o.Ä. auf solchen Sicherungspunkten aufgesetzt und damit ein Wiederanlaufen des Programms initiiert werden.

7.4 WROUT-Funktion

Für ISAM-Dateien kann die sog. „WROUT-Funktion“ (oder auch „Write Immediate“) genutzt werden: sie steuert das Zurückschreiben geänderter Blöcke auf die Platte. Bei eingeschaltetem WROUT wird jeder geänderte Block sofort zurückgeschrieben – damit ist jederzeit Konsistenz der Daten auf Platte und im virtuellen Speicher gewährleistet. Das Einschalten der WROUT-Funktion hat jedoch auch zur Folge, dass auf Grund der erhöhten Zahl von Ein-/Ausgaben die Performance stark sinkt.

Ist die WROUT-Funktion nicht eingeschaltet, erfolgen Ein-/Ausgaben nur dann, wenn der Inhalt des betreffenden Pufferbereichs ersetzt werden muss. Durch dieses verzögerte Zurückschreiben lassen sich z.B. Schreiboperationen auf Platte sparen, wenn im gleichen Block mehrfach geändert wird.

Die WROUT-Funktion kann explizit oder implizit eingeschaltet werden:

- explizit mit dem Operanden WROUT=YES im FILE- oder im FCB-Makroaufruf sowohl für K-ISAM-Dateien als auch für NK-ISAM-Dateien
- explizit mit dem Operanden WRITE-IMMEDIATE=*YES im Kommando ADD-FILE-LINK sowohl für K-ISAM-Dateien als auch für NK-ISAM-Dateien
- explizit mit WROUT=YES beim Anlegen von task-spezifischen Benutzer-ISAM-Pools mit dem Makro CREPOOL: für alle NK-ISAM-Dateien, die in diesem Pool verarbeitet werden
- explizit mit WRITE-IMMEDIATE=YES beim Anlegen von task-spezifischen Benutzer-ISAM-Pools mit dem Kommando CREATE-ISAM-POOL: für alle NK-ISAM-Dateien, die in diesem Pool verarbeitet werden
- implizit, wenn mit dem Makro CREPOOL bzw. dem Kommando CREATE-ISAM-POOL ein hostspezifischer Pool eingerichtet wird: für alle NK-ISAM-Dateien, die in diesem Pool verarbeitet werden.
  Nicht-taskspezifische ISAM-Pools werden standardmäßig mit dem Attribut WRITE-IMMEDIATE=YES angelegt. Es ist jedoch zuläßig, durch explizite Angabe des entsprechenden Parameters, ISAM-Pools anzulegen, in denen Änderungen verzögert auf Platte geschrieben werden.
- implizit mit der Angabe SHARUPD=YES im FILE- oder im FCB-Makroaufruf, wenn eine Datei für Shared-Update-Verarbeitung eröffnet wird (für K-ISAM und NK-ISAM). Bei der Verarbeitung von K-ISAM-Dateien mit SHARUPD=YES wird stets auf Platte zurückgeschrieben. Bei NK-ISAM Dateien ist das Zurückschreiben auf Platte als Standardverarbeitung voreingestellt. Es ist jedoch möglich auch ohne Zurückschreiben auf Platte zu arbeiten.
- implizit mit der Angabe SHARED-UPDATE=*YES im Kommando ADD-FILE-LINK, wenn eine Datei für Shared-Update-Verarbeitung eröffnet wird (für K-ISAM und NK-ISAM).
  Bei der Verarbeitung von K-ISAM-Dateien mit SHARED-UPDATE=*YES wird stets auf Platte zurückgeschrieben. Bei NK-ISAM Dateien ist das Zurückschreiben auf Platte als Standardverarbeitung voreingestellt. Es ist jedoch möglich auch ohne Zurückschreiben auf Platte zu arbeiten.

Ist die WROUT-Funktion implizit eingeschaltet, kann sie vom Benutzer (mit Ausnahme von NK-ISAM-Dateien) nicht ausgeschaltet werden, d.h. WROUT=NO in den Makros FILE oder FCB bzw. WRITE-IMMEDIATE=*NO im Kommando ADD-FILE-LINK ist wirkungslos.

7.5 Kontroll-Lesen (Write Check)

Kontroll-Lesen (Write Check) ist die Überprüfung auf Aufzeichnungsfehler bei Schreiboperationen auf Plattenspeicher. Da dieses Kontroll-Lesen zusätzliche Plattenumdrehungen erfordert, geht es stark zulasten der Performance.

„Kontroll-Lesen“ kann nur für den aktuellen Verarbeitungslauf eingestellt werden, entweder mit dem Operanden WRCHK in den Makros FCB und FILE oder mit dem Operanden WRITE-CHECK=*YES im Kommando ADD-FILE-LINK. Diese Operanden müssen jedes Mal erneut angegeben werden, wenn Kontroll-Lesen erwünscht ist, da diese Option nicht in den Katalogeintrag aufgenommen wird.

7.6 Dateiattribute für die Datensicherung

Die Sicherungsprogramme ARCHIVE und HSMS, in regelmäßigen Sicherungsläufen eingesetzt, erstellen Sicherungskopien von den aktuellen Dateien.

Der Benutzer kann selbst „Sicherungsstufen“ für seine Dateien definieren, indem er im Makro CATAL oder in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES einen „Backup-Level“ über gleichnamige Operanden einstellt.
Backup-Level „A“ bedeutet z.B. „höchste Sicherungsstufe“, die Datei wird bei jedem Sicherungslauf mit gesichert.
Backup-Level „E“ bedeutet dagegen, dass die Datei nur gesichert wird, wenn in HSMS/ARCHIVE die maximale Backup-Klasse „E“ eingestellt ist.
Ob sich die BACKUP-Einstellung (A/.../E) auf die Sicherungshäufigkeit auswirkt, hängt vom Sicherungsverfahren des Rechenzentrums ab.

Falls das Rechenzentrum partielle Sicherungen durchführt, kann der Benutzer auch das Ausmaß einer partiellen Sicherung für seine Dateien festlegen: sowohl im Makro CATAL (Operand LARGE) als auch in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES (Operand SAVED-PAGES) wird angegeben, ob bei einem partiellen Sicherungslauf die gesamte Datei gesichert werden soll, oder – was für große Dateien sinnvoll ist – ob eine partielle Sicherung ausreicht.

Der Benutzer kann verhindern, dass HSMS eine Datei auf eine Hintergrundebene verdrängt, indem er das Dateiattribut MIGRATE auf INHIBITED oder FORBIDDEN setzt. Die Einstellung INHIBITED kann der HSMS-Verwalter umgehen, aber nicht die Einstellung FORBIDDEN.

Der Benutzer kann einstellen, ob eine Datei am HSMS Versions-Backup teilnehmen kann, indem er für das Dateiattribut NUM-OF-BACKUP-VERS einen Wert größer 0 definiert. Der Wert 0 bedeutet keine Teilnahme. Der Wert legt fest, wieviele Datei-Versionen maximal im HSMS Versions-Backup-Archiv gesichert werden sollen. Temporäre Dateien, Dateien auf privater Platte, Banddateien und Dateigenerationen werden vom Versions-Backup nicht unterstützt, d.h. bei diesen Dateien ist nur der Wert 0 möglich.
Die Einstellung erfolgt im Makro CATAL (Operand NUM_OF_BACKUP_VERS) oder in den Kommandos CREATE-FILE und MODIFY-FILE-ATTRIBUTES (Operand NUM-OF-BACKUP-VERS). Ohne explizite Angabe wird bei Dateien, für die Versions-Backup unterstützt wird, der im Systemparameter NUMBACK definierte Wert übernommen.

Für ausführlichere Informationen zu den Dateisicherungen siehe die Handbücher „ARCHIVE“ [9] und „HSMS“ [10], Makros und Kommandos zur Einstellung der Dateiattribute für die Datensicherung sind in den Handbüchern „DVS-Makros“ [1] und „Kommandos“ [3] beschrieben.

7.7 Pubset-Sicherung mit Snapsets

Ein Snapset ist die Sicherung eines SF- bzw. SM-Pubsets. Ein Snapset dient als Pubset-Sicherung der Wiederherstellung verlorener Daten (z.B. durch versehentliches Löschen).

Snapsets werden im laufenden Pubset-Betrieb von der Systembetreuung oder vom HSMS-Administrator erzeugt und später wieder gelöscht. Beim Erzeugen kann der älteste Snapset implizit gelöscht werden.
Die Snapsets können als logische Sicherung aller Dateien und Jobvariablen eines Pubsets benutzt werden. Auch der komplette Pubset kann auf den Stand zum Zeitpunkt des Anlegens des Snapsets zurückgesetzt werden.

Snapsets werden bei ihrer Erzeugung bzw. beim Importieren des Pubsets in Betrieb genommen und stehen dann für den lesenden Zugriff zur Verfügung. Sie ermöglichen die Restaurierung von Einzeldateien und Jobvariablen auf dem Stand des jeweiligen Snapsets.

Die Nutzung von Snapsets ist auch im Shared-Pubset-Betrieb möglich.

Snapsets eines Pubsets sind temporär nicht verfügbar, wenn das Subsystem SHC-OSD zum Zeitpunkt des Pubset-Imports noch nicht aktiv war (insbesondere werden die Snapsets des Home-Pubsets bei Systemstart nicht automatisch in Betrieb genommen). Sobald SHC-OSD aktiv ist und das Kommando SHOW-SNAPSET-CONFIGURATION aufgerufen wird, werden diese Snapsets nachträglich in Betrieb genommen.

Dateien auf dem Snapset

Das Kommando CHECK-SNAPSET-CONFIGURATION dient der Überprüfung bzw. Aktivierung der Snapset-Konfiguration eines importierten Pubsets.

Mit dem Kommando LIST-FILE-FROM-SNAPSET bzw. dem Makro LFFSNAP kann sich der Benutzer über Dateien informieren, die auf einem Snapset gesichert wurden. Er kann sich dabei über alle für ihn zugreifbaren Dateien informieren (wie bei SHOW-FILE-ATTRIBUTES, das Informationen aus dem aktuellen Dateikatalog liefert).

So gesicherte Dateien können mit dem Kommando RESTORE-FILE-FROM-SNAPSET bzw. dem Makro RFFSNAP von dem Snapset restauriert werden. Beim Restore werden einzelne Dateien von dem Snapset in den laufenden Pubset kopiert. Der Vorgang ist vergleichbar mit einem HSMS-Restore aus einem Backup-Archiv.

Es kann ein bestimmter Sicherungsstand (voreingestellt ist die jüngste Snapset-Sicherung) vorgegeben werden. Alternativ können die Dateien unter Einbeziehung aller vorhandenen Snapsets jeweils mit ihrem neuesten Dateistand restauriert werden.

Alle Dateiattribute einer restaurierten Datei werden unverändert von der Originaldatei übernommen (auch Erstellungs- und Änderungsdatum sowie die Schutzattribute). Nur die Allokierung kann von der der Originaldatei abweichen, auch bei Dateien mit physikalischer Allokierung. Dateien auf SM-Pubsets werden auf dem „passendsten “ Volume-Set restauriert. Dieser kann von dem ursprünglichen Volume-Set abweichen.

Einzelne Dateigenerationen können nur mit der gesamten Dateigenerationsgruppe restauriert werden. Dateien auf Privatplatte werden nicht berücksichtigt. Bei migrierten Dateien und Banddateien werden nur die Katalogeinträge restauriert (ohne die Verfügbarkeit der zugehörigen Bänder zu prüfen). Migrierte Dateien und Banddateien können beim Restaurieren nicht umbenannt werden.

Der nichtprivilegierte Benutzer kann die Datei einer fremden Benutzerkennung nur restaurieren, wenn er das Leserecht für die Originaldatei und Eigentumsrechte für die Zieldatei besitzt.

Für bereits vorhandene Dateien muss das Überschreiben durch das Restaurieren explizit zugelassen werden (Operand REPLACE). Für Dateien, die mit Kennwort gegen unberechtigtes Überschreiben geschützt sind, muss das erforderliche Kennwort in der Kennworttabelle des Aufrufers eingetragen sein (siehe ADD-PASSWORD).

Dateien können auch unter einem neuen Namen restauriert werden (Operand NEW-FILE-NAME). Die Umbenennung erfolgt entweder durch Angabe einer anderen Benutzerkennung oder eines Dateinamenspräfix.

Optional können Dateien, die zum Zeitpunkt der Snapset-Erzeugung schreibgeöffnet waren, restauriert werden (Operand RESTORE-OPEN-FILES). Eine so restaurierte Datei hat einen Zustand wie nach einem Systemabsturz. Für eine ISAM-Datei kann ein Verify notwendig werden. Die dauerhaft geöffneten Systemdateien von SRPM, Guards und GCF werden beim Erzeugen eines Snapsets in einen konsistenten Zustand gebracht, der ein späteres Restaurieren erlaubt.

Bei Bedarf kann sich der Aufrufer ein Protokoll der Restore-Verarbeitung nach SYSOUT bzw. SYSLST ausgeben lassen (Operand OUTPUT). Das Protokoll kann entweder alle Dateien oder nur die Dateien, die aus bestimmten Gründen nicht restauriert werden konnten, umfassen (Operand REPORTING).

Jobvariablen auf dem Snapset

Für Jobvariablen greift der gleiche Mechanismus wie bei Dateien (siehe vorherigen Abschnitt).

Unterschiede ergeben sich nur bei den zuständigen Kommandos und/oder Operanden:

- Das Kommando LIST-JV-FROM-SNAPSET bzw. der Makro LJFSNAP informiert über Jobvariablen, die auf einen Snapset gesichert wurden (wie SHOW-JV-ATTRIBUTES, das Informationen über Jobvariablen aus dem aktuellen Dateikatalog liefert).
- Das Kommando RESTORE-JV-FROM-SNAPSET bzw. der Makro RJFSNAP restauriert Jobvariablen auf dem Stand eines bestimmten Snapsets oder auf dem aktuellsten Stand auf Basis aller vorhandenen Snapset. Das Umbenennen erfolgt hier über den Operanden NEW-JV-NAME. Die weiteren Optionen zum Überschreiben (Operand REPLACE) und Steuern der Protokollausgabe (Operanden OUTPUT und REPOR-TING) sind wie beim Restaurieren von Dateien vorhanden.

Snapsets anzeigen

Über vorhandene Snapsets (= existierende Sicherungsstände des Pubsets) kann sich der Benutzer mit dem Kommando SHOW-SNAPSET-CONFIGURATION informieren.

Die Ausgabe erfolgt nach SYSOUT, optional kann sie auch nach SYSLST erfolgen.
Für jeden Snapset werden u.a. das Erstellungsdatum, die Snapset-Id, die Kennung der CCOPY-Session, über die der Zugriff auf den Snapset erfolgen kann, der Name des zugeordneten Snap-Pools und die Kennung des Target-Plattenspeichersystems ausgegeben, falls für die Target-Units ebenfalls Snap-Kopien erstellt wurden (vgl. auch Handbuch „SHC-OSD“ [26]).

Bibliothekselemente restaurieren

Während eine PLAM-Bibliothek mit dem Kommando RESTORE-FILE-FROM-SNAPSET bzw. dem Makro RFFSNAP als gesamte Datei restauriert wird, kann die PLAM-Bibliothek in LMS direkt auf dem Snapset geöffnet werden. Dies ermöglicht den lesenden Zugriff auf die einzelnen Bibliothekselemente und damit das elementweise Restaurieren.

Dateien und Jobvariablen von Snapsets in Backup-Archiv übernehmen

Mit der HSMS-Anweisung BACKUP-FILES können Dateien und Jobvariablen, die auf einen Snapset gesichert wurden, in ein Backup-Archiv übernommen werden. Damit können Dateien und Jobvariablen praktisch „offline“ zum laufenden Pubset-Betrieb in eine längerfristig angelegte Sicherung überführt werden. Näheres siehe Handbuch „HSMS“ [10].

8 Datenträger- und Geräteverwaltung

Welche Geräte stehen Benutzeraufträgen zur Verfügung?

Benutzeraufträge können alle peripheren Geräte mit Ausnahme der Systembetriebsmittel reservieren. Systembetriebsmittel sind:

- Konsolen
- Plattengeräte für gemeinschaftliche Datenträger
- Geräte, die der Operator dem System zugewiesen hat (z.B. Drucker)

In der Regel können sich Benutzeraufträge darauf beschränken, Band- und Plattengeräte zu reservieren. Andere Geräte wie Drucker werden vom System bedient und können über die SPOOL-Schnittstellen angesprochen werden (Kommando PRINT-DOCUMENT bzw. PRNTDOC-Makro, WRLST-Makroaufruf).

8.1 Datenträger und Geräte anfordern

Benutzeraufträge können mit dem FILE-Makroaufruf und den Kommandos CREATE-FILE, IMPORT-FILE, MODIFY-FILE-ATTRIBUTES, ADD-FILE-LINK und SECURE-RESOURCE-ALLOCATION Geräte und Datenträger anfordern. Für Banddateien können die Datenträger auch während der OPEN-Verarbeitung oder beim automatischen Bandwechsel angefordert werden.

Mit den Kommandos CREATE-FILE und IMPORT-FILE werden Geräte und Datenträger angefordert, die für eine bestimmte Datei benötigt werden.

Mit dem SECURE-RESOURCE-ALLOCATION-Kommando lassen sich auch andere Betriebsmittel reservieren, wie z.B. Dateien auf gemeinschaftlichen Datenträgern. Wird mit einem SECURE-RESOURCE-ALLOCATION-Kommando eine Datei reserviert, die sich über mehrere private Datenträger erstreckt, belegt das DVS alle Datenträger und die dafür benötigten Geräte.

In einem Batchauftrag bewirkt das Kommando SECURE-RESOURCE-ALLOCATION darüber hinaus, dass der Auftrag wartet, wenn zum Zeitpunkt des Anforderns einer der angeforderten Datenträger nicht frei ist. Die maximale Wartezeit kann auch explizit durch den WAIT-Operanden im Kommando SECURE-RESOURCE-ALLOCATION angegeben werden (sowohl für Batch- wie für Dialogbetrieb).

Geräte können über den Gerätetyp (in den Kommandos CREATE-FILE, IMPORT-FILE und SECURE-RESOURCE-ALLOCATION) angefordert werden: die Geräteverwaltung wählt in diesem Fall ein beliebiges freies Gerät des angegebenen Typs aus.
Die Typ-Reservierung von Plattengeräten durch das Kommando SECURE-RESOURCE-ALLOCATION kann nur für spezielle Anwendungen (z.B. VOLIN) genützt werden.

Gerätereservierung mit dem mnemotechnischen Gerätenamen ist mit dem Kommando SECURE-RESOURCE-ALLOCATION möglich. Das NDM (Nucleus Device Management, Geräteverwaltung) belegt das Gerät mit dem angegebenen Namen, falls es frei ist. Bei Plattengeräten ist eine solche Belegung allerdings nur für spezielle Anwendungen verwendbar.

Wenn alle angeforderten Geräte frei sind, werden sie für den anfordernden Auftrag reserviert. Haben Sie gleichzeitig Datenträger angefordert, sendet das NDM eine MOUNT-Meldung an der Konsole, d.h. es fordert den Operator auf, die Datenträger einzuhängen. Für Dateien auf privaten Platten müssen alle Platten, die Teile der Datei enthalten, bereitgestellt sein.

Ist eines der angeforderten Geräte nicht frei, werden keine Geräte reserviert. Im Dialogbetrieb wartet der Auftrag nicht, im Batchbetrieb wird der Auftrag in eine Warteschlange eingereiht (falls die Reservierung mit SECURE-RESOURCE-ALLOCATION angefordert wurde). Der Auftrag wird fortgesetzt, wenn alle angeforderten Geräte während der vereinbarten Wartezeit verfügbar werden. Wurden die Geräte mit dem Kommando CREATE-FILE angefordert oder sind nicht alle Geräte verfügbar, wird der Spin-Off-Mechanismus ausgelöst, das DVS verzweigt zum nächsten SET-JOB-STEP- oder LOGOFF-Kommando.

8.2 Datenträger und Geräte freigeben

Für die Datenträger- und Geräte-Freigabe gibt es folgende Möglichkeiten:

- Auftragsende / Auftragsabbruch
- RELEASE (RELTFT-Makroaufruf); die einer Datei zugeordneten Geräte und Datenträger werden freigegeben
- Kommando REMOVE-FILE-LINK
- Kommando SECURE-RESOURCE-ALLOCATION; bei Angabe dieses Kommandos werden alle zuvor reservierten und/oder belegten Datenträger und Geräte freigegeben

Wenn ein Auftrag deaktiviert (Kommando WAIT-EVENT) oder beendet wird (Kommando LOGOFF) oder auch bei Auftragsabbruch werden implizit die reservierten Geräte freigegeben.

RELEASE

Die einer Datei zugeordneten Geräte und Datenträger werden freigegeben, der Eintrag in der TFT gelöscht. Existiert eine HOLD-Sperre, wird der RELEASE allerdings erst nach DROP ausgeführt.

Sind Bänder bzw. Geräte, die der Datei zugeordnet sind, auch noch für andere Dateien bzw. Bänder implizit reserviert, werden sie erst nach Abbau sämtlicher Bezüge frei.

Der Makroaufruf RELTFT ...,KEEP hebt die Reservierung der Datei und ggf. der zugehörigen Bänder auf, Geräte bleiben reserviert.

REMOVE-FILE-LINK

Die einer Datei zugeordneten Geräte und Datenträger werden freigegeben.
Mit REMOVE-FILE-LINK wird ein Eintrag in der TFT gelöscht. Die der Datei zugeordneten Bänder und Bandgeräte werden freigegeben. Existiert eine LOCK-FILE-LINK-Sperre, wird das Kommando REMOVE-FILE-LINK allerdings erst nach UNLOCK-FILE-LINK ausgeführt.

Sind Bänder bzw. Geräte, die der Datei zugeordnet sind, auch noch für andere Dateien bzw. Bänder implizit reserviert, werden sie erst nach Abbau sämtlicher Bezüge frei.

REMOVE-FILE-LINK mit der Angabe RELEASE-DEVICE=NO hebt die Reservierung der Datei und ggf. der zugehörigen Bänder auf, Geräte bleiben reserviert.

SECURE-RESOURCE-ALLOCATION

Werden im SECURE-RESOURCE-ALLOCATION-Kommando (kurz: SEC-RES) keine Operanden angegeben, werden alle für den Auftrag reservierten Geräte und Datenträger freigegeben. Auch bei Anforderung weiterer Geräte oder Datenträger werden zunächst alle reservierten Geräte freigegeben, falls nicht Belegungsbezüge für geöffnete Dateien existieren.

Dieses Verhalten hat zur Folge, dass nach Abweisung eines SEC-RES-Kommandos sämtliche vorhergehenden Reservierungen zurückgesetzt sind. In folgenden Fällen bleiben sie jedoch bestehen:

- Die Abweisung erfolgt auf Grund eines Syntaxfehlers
- Die Abweisung erfolgt, weil eine Banddatei geöffnet ist
- Die Abweisung erfolgt, weil ein TFT-Eintrag für die Datei mit LOCK-FILE-LINK gesperrt ist

Das SECURE-RESOURCE-ALLOCATION-Kommando wird abgewiesen, wenn eine Banddatei geöffnet oder ein TFT-Eintrag für die Datei mit LOCK-FILE-LINK gesperrt ist.

8.3 Eingriff durch den Operator

Wenn ein Auftrag Geräte und Datenträger erfolgreich angefordert hat, wird ein Eingriff des Operators notwendig: Die Datenträger sind in die entsprechenden Geräte einzuhängen.

Diese Tätigkeit steuert das NDM (Nucleus Device Management = Geräteverwaltung) mit Meldungen, die den Operator auffordern, einen Datenträger auf einem bestimmten Gerät einzuhängen (MOUNT-Meldungen). Jede dieser Meldungen enthält die Archivnummer des betreffenden Datenträgers und den mnemotechnischen Gerätenamen des Geräts, das bereits für den anfordernden Auftrag reserviert ist.

Der Operator kann den Datenträger auf dem angeforderten Gerät oder einem Gerät gleichen Typs einhängen oder die Anforderung ablehnen. Der Benutzer erhält eine entsprechende Meldung auf der Datenstation, wenn er für seinen Auftrag das Kommando MODIFY-JOB-OPTIONS LOGGING=*PAR(LIST=*YES) eingegeben hat.

8.4 Verwaltung von privaten Platten

Der Benutzer sieht nur die Belegung des Datenträgers. Das dazu benötigte Gerät wird vom System verwaltet. Es gibt keine Reservierung für das Gerät.

Alle Belegungsfunktionen, die infolge von DVS-Aufrufen ablaufen, werden als mehrbenutzbare Belegungsanforderungen interpretiert.

Das Kommando SECURE-RESOURCE-ALLOCATION bietet die Möglichkeit der mehrbenutzbaren und exklusiven Reservierung eines privaten Datenträgers.

Steuerungs- und Überwachungsarten von Datenträgern können über Operatorkommandos eingestellt und verändert werden.

Die Geräteverwaltung bietet die automatische Online-Überwachung der benutzten Datenträger.

Die Datenträger-Überwachung unterstützt und überwacht u.a. das Montieren der Datenträger, aber auch den Zugriff auf die Datenträger während der Dateiverarbeitung.

8.5 Bänder / Bandgeräte

Jeder Zugriff auf eine Banddatei setzt voraus, dass die zugehörigen Datenträger (Magnetbänder, Magnetbandkassetten (MBK)) und Geräte (Bandgeräte, MBK-Geräte) verfügbar sind. Bänder und Bandgeräte können vom Benutzer explizit mit den Kommandos CREATE-FILE, ADD-FILE-LINK und SECURE-RESOURCE-ALLOCATION bzw. mit dem Makro FILE angefordert werden, oder implizit zum OPEN-Zeitpunkt einer Datei bzw. zum Zeitpunkt des Bandwechsels, wenn das Folgeband auf einem anderen Gerät montiert werden soll.

Bandanforderungen über die Kommandos SECURE-RESOURCE-ALLOCATION und ADD-FILE-LINK sowie über den Makro FILE lösen an der Konsole eine PREMOUNT-Meldung aus, es sei denn, Bänder werden offline reserviert.
Bandanforderungen beim OPEN lösen an der Konsole eine MOUNT-Meldung aus. MOUNT=0 bedeutet, dass noch keine PREMOUNT-Meldung an der Konsole ausgegeben wird, das Gerät ist offline reserviert. Erst wenn das Gerät benötigt wird, gibt das DVS die MOUNT-Meldung aus, die der Operator beantworten muss; das Gerät wird dem anfordernden Auftrag zugewiesen.

Standardmäßig gibt das DVS keine MOUNT-Meldungen aus, wenn die angeforderten Bänder bereits montiert sind. In diesem Fall erfolgt eine implizite Gerätezuweisung.
Bei Verarbeitung von MF/MV-Sets gibt das BS2000 PREMOUNT-Meldungen aus, die es dem Operator ermöglichen, Folgebänder rechtzeitig zu montieren.

Der Operator kann die MOUNT-Meldung für ein bestimmtes Gerät akzeptieren oder ein anderes Gerät vorschlagen. Bestätigt er die MOUNT-Meldung, wird das angeforderte Gerät dem Auftrag zugewiesen.

Wenn Geräte implizit (z.B. über den Gerätetyp) reserviert werden, kann der belegende Datenträger beim Auftreten von Gerätefehlern auf ein anderes Gerät ummontiert werden, bei expliziter Reservierung ist dies nicht möglich.

Jedes Ummontieren sollte der Geräteverwaltung mitgeteilt werden. Das System schlägt dem Operator dann ein anderes Gerät vor. Die Geräteverwaltung übernimmt die nach dem Ummontieren notwendige Positionierung des Bandes, sodass eine ordnungsgemäße Weiterverarbeitung des Bandes gesichert ist.

8.5.1 Gerätereservierung / Gerätebelegung

Magnetbandgeräte können über Makro- oder Kommandoaufrufe reserviert bzw. belegt werden.

Makroaufruf

Wann die Gerätebelegung stattfindet, kann der Benutzer steuern:

- Beim FILE-Makroaufruf finden so viele Gerätereservierungen statt, wie Bänder im MOUNT-Operanden angegeben wurden (Standard: ein Gerät wird reserviert). Für MOUNT=0 findet keine Reservierung statt. Sie muss ggf. zum OPEN-Zeitpunkt erfolgen
- Im Rahmen der OPEN-Bearbeitung findet eine Gerätebelegung statt, wenn zuvor kein Gerät mit FILE belegt wurde

Kommandoaufruf

Bandgeräte kann der Benutzer durch die Kommandos ADD-FILE-LINK (mit den Operanden VOLUME-LIST=*CATALOG und VOLUME-LIST=*TAPE-SET) und SECURE-RESOURCE-ALLOCATION sowie durch eine OPEN-Verarbeitung für Banddateien belegen. Wann die Gerätebelegung stattfindet, kann der Benutzer steuern:

- Beim Kommando ADD-FILE-LINK werden so viele Geräte belegt, wie im Operanden NUMBER-OF-PREMOUNTS angefordert wurden.
- Auch im SECURE-RESOURCE-ALLOCATION-Kommando kann der Benutzer festlegen, wie viele Geräte für den Auftrag reserviert werden sollen: Wenn er über den FILE-Operanden Geräte implizit anfordert, kann er mit MOUNT=n angeben, wie viele Geräte benötigt werden. Bei MOUNT=0 findet die Gerätereservierung ggf. erst zum OPEN-Zeitpunkt statt.
- Im Rahmen der OPEN-Bearbeitung findet eine Gerätebelegung statt, wenn zuvor kein ADD-FILE-LINK oder SECURE-RESOURCE-ALLOCATION-Kommando gegeben wurde oder wenn mit einem ADD-FILE-LINK- bzw. SECURE-RESOURCE-ALLOCATION-Kommando eine Gerätebelegung für einen nicht passenden Volume-Typ erfolgt ist (es wird ein weiteres Gerät belegt).

Bei den MBK-Geräten sind nur identische Angaben des Gerätetyps bei den Kommandos SECURE-RESOURCE-ALLOCATION und ADD-FILE-LINK miteinander verträglich.

Eine UNIT-Reservierung hat den Nachteil, dass im Fall eines permanenten Hardware-Fehlers kein Umhängen des Bandes auf ein anderes freies Bandgerät möglich ist: die Bandverarbeitung für den auf diesem Gerät montierten Datenträger wird abnormal beendet (SHOW-TAPE-STATUS zeigt ACTION=CANCELLED). Daher ist zu empfehlen, entweder Geräte explizit über ihren Typ oder implizit durch Reservierungen der notwendigen Bänder zu belegen.

Wurde das angeforderte Band vom Operator nicht auf dem per UNIT reservierten Gerät bereitgestellt, muss es auf das mit UNIT bezeichnete Gerät ummontiert werden; es wird kein neues Band bereitgestellt. Ging der Bandanforderung keine UNIT-Reservierung voraus, kann das Band auf jedem beliebigem Gerät montiert werden, das die angeforderte Schreibdichte unterstützt.

8.5.2 VSN-Mehrdeutigkeit

Sind in einem Rechenzentrum gleichzeitig mehrere Bänder derselben VSN montiert, wird der Operator bei einer Anforderung durch den Benutzer zum Zuweisen des richtigen Bandes aufgefordert.

Seitens der Geräteverwaltung wird garantiert, dass gleichzeitig immer nur ein Datenträger derselben VSN beschrieben werden kann. Parallel dazu können jedoch eine oder mehrere „SPECIAL“-Anwendungen (INIT, VOLIN,...) laufen, die mit derselben VSN arbeiten.
Die Geräteverwaltung hat jedoch keine Möglichkeit, den Benutzer vor Fehleingriffen des Operators zu schützen, d. h., dass evtl. nach einem INOP vom Operator ein falsches Band (gleicher VSN) zugewiesen werden kann und dadurch die Bandverarbeitung auf einem zweiten Datenträger fortgesetzt wird.

VSN-Mehrdeutigkeit kann z.B. auftreten bei der Verarbeitung von Bändern verschiedener Kunden in Auftragsrechenzentren.

8.5.3 Bandwechsel bei Folgebandverarbeitung und PREMOUNT-Anweisung

Folgebänder können vorab auf jedem physikalisch kompatiblen Bandgerät montiert werden, es sei denn, ein Bandgerät wurde über das Kommando SECURE-RESOURCE-ALLOCATION UNIT=mn belegt. Dann muss auch das Folgeband auf demselben Gerät verarbeitet werden wie der Vorgänger.

Erkennt das DVS auf Grund des Makros FILE bzw. des Kommandos ADD-FILE-LINK oder im Katalogeintrag einer Datei, dass diese auf mehreren Bändern gespeichert ist, wird nach der Kennsatzverarbeitung des aktuellen Bandes ein Hinweis für den Operator ausgegeben, der ihm mitteilt, dass ein Folgeband montiert werden kann.

Ist ein Bandwechsel erforderlich, weil sich die Datei auf einem Folgeband fortsetzt oder während der Positionierung eines MF/MV-Sets auf dem aktuellen Band nicht gefunden wird, wird das aktuelle Band an den Bandanfang zurückgespult und entladen. Ist das Folgeband bereits auf einem anderen Gerät montiert, wird es vom DVS ohne Operatoreingriff benutzt. Dies geschieht auch mit dem ersten Band einer Datei im Fall einer Vorabmontierung!

Ist das benötigte Folgeband nicht montiert, erhält der Operator eine MOUNT-Meldung. Dabei wird ihm vom System vorgeschlagen, das Folgeband auf dem Gerät zu montieren, auf dem sich das fortzusetzende Band befindet. Wird das Folgeband auf einem anderen Gerät montiert und dieses zugewiesen, wird dies von der Geräteverwaltung akzeptiert.
In diesem Fall wird das bisher belegte Gerät freigegeben.

8.5.4 Wechsel von Magnetbandgeräten bei Gerätefehlern

Werden Bänder über die Archivnummer, Geräte über den Gerätetyp oder beide implizit über die zu verarbeitende Datei angefordert, kann der Operator beim Auftreten von Gerätefehlern das Band auf ein anderes Gerät des gleichen Gerätetyps ummontieren.

Wurde ein Gerät explizit über seinen mnemotechnischen Gerätenamen reserviert (Operand UNIT im Kommando SECURE-RESOURCE-ALLOCATION), kann der darauf montierte Datenträger nicht auf ein Ersatzgerät ummontiert werden.

Im Fehlerfall bedeutet dies, dass bei expliziter Gerätebelegung (mit UNIT) die Bandverarbeitung abnormal beendet wird. Der Operator weist die PREMOUNT-Meldung ab, das Band wird implizit „gecancelt“. Dieser CANCEL-Zustand kann nur durch Abbau der Bandbelegung wieder aufgehoben werden.

8.6 Informationsausgabe über Betriebsmittel

Die Geräteverwaltung bietet dem Benutzer die Möglichkeit, sich mit dem Makro NKDINF und den unten aufgeführten Kommandos über die aktuellen Betriebsmittelbelegungen seines Auftrags zu informieren, das heißt über Geräte, Datenträgereigenschaften, Reservierungen usw.

Wird der Makro NKDINF verwendet, können mit dessen Operanden die Information bzgl. bestimmter Gerätefamilien usw gesteuert werden. Außerdem bietet der Operand RECORD zu jedem Operanden eine DSECT für den Ausgabebereich an.

Werden die verschiedenen NDM-Kommandos verwendet, zeigt die nachfolgende Tabelle, mit welchen Kommandos die gewünschten Informationen angefordert werden können.

Kommando	Information über
SHOW-DEVICE-CONFIGURATION	Anlagenkonfiguration
SHOW-DEVICE-DEPOT	Zuordnung von Bandgeräten zu Lagerorten
SHOW-DEVICE-STATUS	Belegungszustand, Geräteüberwachung
SHOW-DISK-DEFAULTS	Standardwerte für Plattenparameter
SHOW-DISK-STATUS	Plattenbelegung und Plattenparameter
SHOW-MOUNT-PARAMETER	Montier-Vorgaben
SHOW-RESOURCE-ALLOCATION	Betriebsmittelbelegungen, offene Operator-Aktionen
SHOW-RESOURCE-REQUESTS	Zustand von Gerätewarteschlange und Collector-Task
SHOW-TAPE-STATUS	Bandbelegung, Bandüberwachung

Tabelle 25: Kommandos zur Informationsausgabe über Betriebsmittel

Welche Bedeutung die Ausgabespalten dieser SHOW-Kommandos haben, zeigt die Tabelle im Handbuch „Kommandos“ [3].

9 Dateien auf Magnetband oder MBK

9.1 Definitionen

Im Folgenden wird der Begriff „Band“ sowohl für Magnetbänder als auch für Magnetbandkassetten (MBK) verwendet. Haben Magnetbänder und Magnetbandkassetten unterschiedliche Eigenschaften, wird gesondert darauf hingewiesen.

9.1.1 Datei, Dateiabschnitt und Dateimenge

Ein Dateiabschnitt ist der auf einem Band aufgezeichnete Teil einer Datei.

Wird eine Datei vollständig auf einem Band aufgezeichnet, besteht sie nur aus einem Abschnitt.
Besteht eine Datei aus mehreren Abschnitten, muss der erste Abschnitt der Datei der letzte oder einzige Abschnitt auf einem Band sein; jeder „Zwischenabschnitt“ muss als Einziger auf einem Band gespeichert sein; der letzte Dateiabschnitt muss der Erste oder Einzige auf einem Band sein. Alle Dateiabschnitte sind – beginnend bei 1 – durch ihre fortlaufende Nummerierung identifizierbar (DIN 66029).

Eine Dateimenge setzt sich zusammen aus einer oder mehreren Dateien mit gemeinsamen Dateimengenkennzeichen. Die Dateien der Dateimenge sind über eine fortlaufende Nummerierung – beginnend bei 1 – identifizierbar. Die Dateien einer Dateimenge sind fortlaufend über ein oder mehrere Bänder aufgezeichnet (DIN 66029). Dateimengenkennzeichen ist die Archivnumer (VSN) des ersten Bandes der Bandmenge.

Alle Dateien einer Dateimenge müssen mit denselben Kennsatz- und Code-Eigenschaften sowie Dateischutzeigenschaften erstellt werden.

9.1.2 Bandmenge, Bandfolge und Bandeigentümer

Eine Bandmenge umfasst die Bänder, auf denen eine Dateimenge gespeichert ist. Eine Bandmenge darf nur eine Dateimenge enthalten (DIN 66029). Die „Ein-Band-Datei“ ist daher als Sonderfall einer „Dateimenge auf einem Band “ zu betrachten.

Eine Bandfolge (= Volume Series) ist die Menge der Bänder, die die Datenträgerliste des Katalogeintrags einer Datei bilden. Sie muss nicht identisch sein mit der Bandmenge. Die Bandfolge/Datenträgerliste kann auch Datenträger enthalten, die von der Datei (noch) nicht belegt werden. Allerdings muss die Datei auf dem ersten Band der Bandfolge beginnen. Die Reihenfolge der Bänder entspricht der Reihenfolge der Dateiabschnitte.

Ist die Datei Bestandteil einer Dateimenge, können auf dem ersten Band ihrer Bandfolge auch andere Dateien gespeichert sein; sie muss jedoch noch auf diesem Band beginnen.

Als Bandeigentümer gilt der Benutzer, unter dessen Kennung die erste Datei auf dem Band erstellt wird. Beim Erstellen der ersten Datei wird die Benutzerkennung als „Eigentümerkennzeichen“ in das entsprechende Feld des VOL1-Kennsatzes eingetragen. Alle Bänder einer Bandmenge sollten im VOL1-Kennsatz das gleiche Eigentümerkennzeichen enthalten.

Mehrbenutzbarkeit von Bändern/Banddateien

Die „Mehrbenutzbarkeit“ eines Bandes richtet sich danach, ob die erste Datei auf dem Band mehrbenutzbar ist. Wird eine Banddatei mit dem Makro FILE oder dem Kommando CREATE-FILE erstellt, wird sie implizit als mehrbenutzbar katalogisiert (Ausgabefeld USER-ACC(ESS)=ALL-USERS). Auch das Band ist dann mehrbenutzbar. Bänder und Banddatei können für fremden Zugriff gesperrt werden, wenn die Datei

- zuerst mit dem CATAL-Makro katalogisiert wird (implizit USER-ACC=OWNER-ONLY) und anschließend erst der FILE-Makro mit den Operanden DEVICE und VOLUME abgesetzt wird
- mit einem FILE-Makro (Operanden DEVICE und VOLUME) katalogisiert und anschließend mit einem CATAL-Makro (Operanden STATE=UPDATE und SHARE=NO) für „nicht-mehrbenutzbar“ erklärt wird. Der CATAL-Makro muss vor dem ersten Öffnen der Datei wirksam werden
- zuerst mit dem Kommando CREATE-FILE katalogisiert wird und dabei für USER-ACCESS=*OWNER-ONLY sowie SUPPORT=*NONE eingetragen und anschließend erst das Kommando MODIFY-FILE-ATTRIBUTES mit den Operanden VOLUME und DEVICE-TYPE abgesetzt wird
- mit dem Kommando CREATE-FILE katalogisiert und anschließend mit dem Kommando MODIFY-FILE-ATTRIBUTES über den Operanden USER-ACCESS=*OWNER-ONLY für „nicht-mehrbenutzbar“ erklärt wird. Das Kommando MODIFY-FILE-ATTRIBUTES muss vor dem ersten Öffnen der Datei wirksam werden

9.1.3 Freies Band (Scratch Tape)

Ein Band wird als „freies Band“ betrachtet, wenn es entweder noch nicht beschrieben wurde (also keine Dateien enthält) oder wenn die Sperrfrist der ersten Datei auf dem Band abgelaufen und Schreibzugriff erlaubt ist.

Nach Abschluss der Verarbeitung wird das Band archiviert, und die Datei auf dem Band hat alle vereinbarten Schutzattribute.

9.2 Kennsätze

- DIN-Normen
- Kennsatz-Typen (Standardkennsätze)
  - System-Kennsätze
  - Benutzer-Kennsätze
  - Reihenfolge der Kennsätze / Kennsatzgruppen
- Nicht-Standardkennsätze / kennsatzlose Banddateien

9.2.1 DIN-Normen

Die Aufzeichnung von Daten auf Magnetbändern/Magnetbandkassetten wird über DIN-Normen geregelt. Dadurch ist Datenaustausch über Bänder zwischen verschiedenen Systemen möglich.

Vom BS2000 unterstützte DIN-Normen:

- **DIN 66029, Austauschstufen 1-3**
  Kennsätze und Dateianordnung auf Magnetbändern für den Datenaustausch
- **DIN 66229, Ausbaustufe**
  Kennsätze und Dateianordnung auf Magnetbandkassetten für den Datenaustausch; Spurkennsätze werden nicht erzeugt

Die o.g. Normen beziehen sich nur auf die logische Anordnung von Dateien und Kennsätzen auf Bändern, nicht auf die physikalischen Aufzeichnungsverfahren und Eigenschaften von Bändern; diese sind in weiteren Normen geregelt. Die Normen DIN 66229 und DIN 66029 beziehen sich nur auf den Teil des Bandes zwischen Bandanfangsmarke (BOT= Begin of Tape) und der letzten Doppel-Abschnittsmarke (Double Tape Mark), die einer Band- oder Dateiende-Kennsatzgruppe folgt.

Ein Benutzer kann für seine Bänder/Banddateien die genormten Standardkennsätze und/oder eigene Benutzer-Kennsätze verwenden. Er kann auch auf Kennsätze verzichten.

Die Austauschstufen von DIN 66029 beschreiben folgende Einschränkungen:

- Austauschstufe 1: eine Bandmenge enthält nur eine Datei; die Sätze der Datei haben feste Länge
- Austauschstufe 2: eine Bandmenge enthält nur Dateien mit Sätzen fester Länge
- Austauschstufe 3: eine Bandmenge enthält nur Dateien mit Sätzen fester oder variabler Länge

Die DIN-Normen werden von der Zugriffsmethode SAM voll erfüllt, die Zugriffsmethoden BTAM und UPAM sind blockorientierte Zugriffsmethoden und unterstützen daher keine satzweise Dateiverarbeitung; UPAM unterstützt auch keinen Bandwechsel.

> ¡ Die folgenden Beschreibungen beziehen sich auf Bänder und Banddateien mit Standardkennsätzen. Auf Nicht-Standardkennsätze bzw. kennsatzlose Verarbeitung wird gesondert verwiesen.

9.2.2 Kennsatz-Typen (Standardkennsätze)

Position des Kennsatzes	Kennsatztyp	Kennsatz-Name	zulässige Kennsätze
Bandanfang	Bandanfangs-Kennsatz (Volume Header Label)	VOL	VOL1
	Benutzer-Bandanfangs-Kennsatz (User Volume Header Label)	UVL	UVL1 ... UVL9
Dateianfang oder Dateiabschnittsanfang	Dateianfangs-Kennsatz (File Header Label)	HDR	HDR1 ... HDR3
	Benutzer-Dateianfangs-Kennsatz (User File Header Label)	UHL	UHL0 ... UHL255
Ende eines Dateiabschnitts (mit Ausnahme des letzten Dateiabschnitts)	Bandende-Kennsatz (End of Volume Label)	EOV	EOV1 ... EOV3
	Benutzer-Bandende-Kennsatz (User End of Volume Label)	UTL	UTL0 ... UTL255
Dateiende oder Ende des letzten Dateiabschnitts	Dateiende-Kennsatz (End of File Label)	EOF	EOF1 ... EOF3
	Benutzer-Dateiende-Kennsatz (User Trailer Label)	UTL	UTL0 ... UTL255

Tabelle 26: Kennsatz-Typen (Standardkennsätze)

9.2.2.1 System-Kennsätze

Über die Makros FILE oder FCB (Operand LABEL) bzw. über das Kommando ADD-FILE-LINK (Operand LABEL-TYPE) wird festgelegt, dass die Banddatei mit Standardkennsätzen erstellt werden soll: der bei den Operanden angegebene Wert bezeichnet die Austauschstufe bzgl. DIN 66029.
Bild 7 (Einbanddatei: Eine Datei auf einem Band) zeigt den Aufbau einer Banddatei mit Standardkennsätzen.

Bandanfangs-Kennsatz (VOL1)

Der erste Block auf einem Magnetband ist der Bandanfangs-Kennsatz VOL1; er darf nur am Bandanfang verwendet werden. Der VOL1-Kennsatz enthält das Datenträgerkennzeichen (= VSN/Archivnummer) und das Eigentümerkennzeichen sowie Kennzeichen, ob das Band mehrbenutzbar ist und welcher DIN-Norm Band- und Datei-Kennsätze entsprechen (Normvermerk).

DIN 66029 erlaubt bis zu 9 Bandanfangs-Kennsätze, das BS2000 schreibt außer VOL1 keine weiteren Bandanfangs-Kennsätze. Werden Nicht-BS2000-Bänder mit mehr als einem Bandanfangs-Kennsatz gelesen, ignoriert das DVS alle Bandanfangs-Kennsätze außer VOL1.

Zur Kennsatzverarbeitung siehe auch Kapitel „Kennsatz-Verarbeitung beim Eröffnen von Banddateien“.

Bandende-Kennsätze (EOV)

Die Bandende-Kennsätze kennzeichnen bei Dateien, die sich über mehrere Bänder erstrecken, das Ende eines Bandes und damit das Ende eines Dateiabschnitts. Dem letzten Datenblock eines Dateiabschnitts folgt zunächst eine Abschnittsmarke, dann die Bandende-Kennsätze und als Abschluss eine Doppel-Abschnittsmarke.

DIN 66029 erlaubt bis zu 9 Bandende-Kennsätze (EOV1, ..., EOV9), das BS2000 unterstützt jedoch nur maximal drei (EOV1, EOV2, EOV3); d.h. die Kennsätze EOV4-EOV9 werden beim Lesen ignoriert; geschrieben werden nur EOV1-EOV3.

Zur Kennsatzverarbeitung siehe auch Kapitel „EOV-Verarbeitung“.

Dateianfangs-Kennsätze (HDR)

Jeder Datei werden Dateianfangs-Kennsätze vorangestellt. Sie dienen der Kennzeichnung der Datei und enthalten die Dateimerkmale, haben also eine ähnliche Funktion wie der Eintrag im F1-Kennsatz einer Privatplatte.

DIN erlaubt bis zu 9 Dateianfangs-Kennsätze (HDR1, ..., HDR9), das BS2000 unterstützt jedoch maximal drei Kennsätze (HDR1, HDR2, HDR3). D.h. die Kennsätze HDR4-HDR9 werden beim Lesen ignoriert und vom BS2000 nicht geschrieben.

Den Dateianfangs-Kennsätzen folgt immer eine Abschnittsmarke. Erstreckt sich eine Datei über mehrere Bänder (Mehrbanddatei), beginnt jeder Dateiabschnitt mit Dateianfangs-Kennsätzen, die dem Bandanfangs-Kennsatz folgen.

Zur Kennsatzverarbeitung siehe auch Kapitel „Kennsatz-Verarbeitung beim Eröffnen von Banddateien“.

Dateiende-Kennsätze (EOF)

Dem letzten Datenblock einer Datei folgen immer – nach einer Abschnittsmarke – die Dateiende-Kennsätze, die wie die Dateianfangs-Kennsätze der Kennzeichnung der Datei und der Überwachung von Dateizugriffen dienen.

DIN 66029 erlaubt bis zu 9 Dateiende-Kennsätze (EOF1, ..., EOF9), das BS2000 unterstützt jedoch nur die Kennsätze EOF1-EOF3; d.h. EOF4-EOF9 werden beim Lesen ignoriert und vom BS2000 nicht geschrieben.

Den Dateiende-Kennsätzen folgt eine Abschnittsmarke, wenn das Band anschließend noch weitere Dateien enthält. Ist die Datei die Letzte auf dem Band, folgt den Dateiende-Kennsätzen eine Doppel-Abschnittsmarke.

Zur Kennsatzverarbeitung siehe auch Kapitel „CLOSE-Verarbeitung“.

9.2.2.2 Benutzer-Kennsätze

Benutzer-Kennsätze müssen im Benutzerprogramm geschrieben/ausgewertet werden. Benutzerprogrammroutinen zum Lesen oder Schreiben von Benutzer-Kennsätzen werden über Ausgänge des EXLST-Makroaufrufs angestoßen: OPENV, LABGN, LABEOV und LABEND.

Die Benutzer-Kennsatzgruppen folgen jeweils den entsprechenden System-Kennsatzgruppen. Wie die System-Kennsätze sind sie durch den Kennsatznamen in den ersten Bytes des Kennsatzes identifizierbar: UVL, UHL und UTL. Es gibt jedoch nur 3 Arten von Benutzer-Kennsätzen: Band- und Dateianfangs-Kennsätze (UVL/UHL) und allgemeine Ende-Kennsätze (UTL).

Benutzer-Bandanfangs-Kennsätze (UVL)

Es können maximal 9 eigene Bandanfangs-Kennsätze (UVL1,...,UVL9) geschrieben werden. Diese stehen zwischen dem VOL1-Kennsatz und den HDR-Kennsätzen.
Die Programmroutine zum Lesen oder Schreiben von UVL-Kennsätzen wird über den EXLST-Ausgang OPENV angestoßen.

Benutzer-Dateianfangs-Kennsätze (UHL)

Es können maximal 256 eigene Dateianfangs-Kennsätze geschrieben werden
(UHL0, ..., UHL255), die dem letzten HDR-Kennsatz folgen müssen. Die Programmroutine zum Lesen/Schreiben von UHL-Kennsätzen wird über den EXLST-Ausgang LABGN angestoßen.

Ende-Kennsätze (UTL)

Benutzer-Bandende- und Dateiende-Kennsätze sind identisch aufgebaut; die Bandende-Kennsätze folgen den EOV-Kennsätzen, die Dateiende-Kennsätze den EOF-Kennsätzen. Es können maximal 256 Band- und 256 Dateiende-Kennsätze geschrieben werden
(UTL0, ..., UTL255). Die entsprechenden Programmroutinen werden über die EXLST-Ausgänge LABEND (für Dateiende) oder LABEOV (für Bandende) automatisch angestoßen.

9.2.2.3 Reihenfolge der Kennsätze / Kennsatzgruppen

System- und Benutzer-Kennsätze müssen innerhalb eines Bandes und innerhalb einer Datei immer in definierter Anordnung aufeinander folgen. Innerhalb von Kennsatzgruppen dürfen keine Abschnittsmarken stehen. Jede Kennsatzgruppe muss vollständig auf dem Band stehen, das den ersten Kennsatz dieser Gruppe enthält.

Alle Kennsatzgruppen beginnen mit dem Kennsatz-1 (HDR1, EOF1, EOV1, UVL1), mit Ausnahme von UHL (UHL0) und UTL (UTL0). Diesem ersten Kennsatz folgen alle weiteren Kennsätze der entsprechenden Gruppe, wobei die Kennsatznummern lückenlos aufsteigend sein müssen.

Die Zahl der Datei- oder Bandende-Kennsätze (EOF/EOV) muss gleich der der Dateianfangs-Kennsätze (HDR) sein.

Enthält ein Band oder eine Datei (bzw. eine Band- oder Dateimenge) sowohl System- als auch Benutzer-Kennsätze, folgt jeweils die Benutzer-Kennsatzgruppe der entsprechenden System-Kennsatzgruppe.

Bei den Benutzer-Kennsätzen wird die Zahl der Ende-Kennsätze (UTL) nicht durch die Zahl der Dateianfangs-Kennsätze (UHL) festgelegt.

9.2.3 Nicht-Standardkennsätze / kennsatzlose Banddateien

Das BS2000 erlaubt auch die Verarbeitung von Bändern/Banddateien, die keine Standardkennsätze enthalten. Auch die Verarbeitung solcher Dateien wird über die Makros FILE und FCB (Operand LABEL) sowie das Kommando ADD-FILE-LINK (Operand LABEL-TYPE) gesteuert. Die Angabe LABEL=NO bzw. LABEL-TYPE=*NO bedeutet, dass die zu verarbeitende Datei ohne Kennsätze erstellt wurde, die Angabe LABEL=NSTD bzw. LABEL-TYPE= *NON-STD, dass sie zwar Kennsätze enthält, jedoch keine Standardkennsätze. Solche Nicht-Standardkennsätze müssen in speziellen EXLST-Routinen des Benutzerprogramms erstellt/ausgewertet werden; Sie brauchen dabei keinerlei Form einzuhalten.

9.3 Dateianordnung auf Magnetbändern und MBK

Der „klassische Fall“ der Magnetbanddatei ist eine einzige Datei auf einem Band. Das BS2000 unterstützt jedoch auch die Verarbeitung von Mehrbanddateien und Mehrdateibändern (= File Sets) sowie von MF/MV-Sets (= Multifile-/Multivolume-Sets), bei denen eine Dateimenge auf einer Bandmenge gespeichert ist. Im Folgenden werden Dateien mit System-Kennsätzen beschrieben, Benutzer-Kennsätze bleiben unbeachtet.

Für die Anordnung der Dateien auf Magnetbändern und MBK gelten dieselben Regeln.

Die Dateiverwaltung des BS2000 hat im Zusammenhang mit der Anordnung von Dateien auf Magnetbändern die Aufgabe, die Erstellung, den Zugriff auf und die Erweiterung von Dateien vorzubereiten und zu überwachen. Sie haben mit den Makros FILE, FCB oder CATAL sowie den Kommandos ADD-FILE-LINK, CREATE-FILE und MODIFY-FILE-ATTRIBUTES die Möglichkeit, die Aufgaben der Dateiverwaltung problemorientiert zu steuern.

Das DVS unterstützt die Verwaltung von

- einer Datei auf einem Band
- einer Datei auf mehreren Bändern
- mehreren Dateien auf einem Band
- mehreren Dateien auf mehreren Bändern
- verschiedenen Versionen einer Datei
- Dateigenerationen

9.3.1 Einbanddatei: Eine Datei auf einem Band

Eine „Einbanddatei“ (= Single Volume File) ist eine Datei, die auf einem einzigen Band gespeichert ist.

Aufbau einer Einbanddatei

Dem Bandanfangs-Kennsatz VOL1 folgen die Dateianfangs-Kennsätze, die vom ersten Datenblock durch eine Abschnittsmarke getrennt sind. Die Datenblöcke können Standardformat haben (PAM-Seiten: 2048 Byte – mit und ohne PAM-Schlüssel) oder Nicht-Standardformat (kein PAM-Schlüssel, variable Blocklänge; siehe Abschnitt „Blockformate für Banddateien“). Auch die Dateiende-Kennsätze sind von den Datenblöcken durch eine Abschnittsmarke getrennt. Den Bandabschluss bildet immer eine Doppel-Abschnittsmarke.

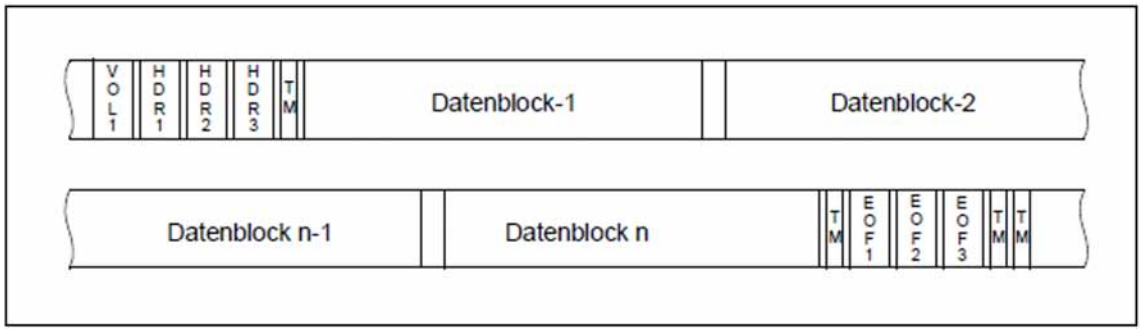


Bild 7: Eine Datei auf einem Band

Erstellen einer Einbanddatei

Im Makro FILE werden mit den Operanden DEVICE und VOLUME, im Kommando CREATE-FILE werden mit den Operanden DEVICE-TYPE und VOLUME der Gerätetyp für den Datenträger und der Datenträger definiert. Mit dem Operanden LINK im Makro FILE bzw. dem Operanden LINK-NAME im Kommando ADD-FILE-LINK wird ein Eintrag in der Task File Table erzeugt. Man kann einen bestimmten Datenträger (VOLUME=VSN) anfordern oder sich vom Operator einen Datenträger zuweisen lassen (VOLUME=PRIVATE bzw. VOLUME=*ANY). Werden keine weiteren Angaben zu LABEL (bzw. LABEL-TYPE) usw. gemacht, wird eine Standard-Banddatei erzeugt, derzeit entsprechend DIN 66029, Austauschstufe 3. Die Datei erhält damit standardmäßig drei HDR-Kennsätze (HDR1-3) und drei EOF-Kennsätze (EOF1-3).

Will ein Benutzer eigene Kennsätze schreiben, muss er im EXLST-Makroaufruf die entsprechenden Kennsatzroutinen definieren. Das System verzweigt automatisch in diese Routinen, wenn die jeweils vorausgehenden System-Kennsätze geschrieben sind.

Bezüglich der übrigen Dateieigenschaften, die über Makro oder Programm definiert werden (Zugriffsmethode, Satzformat usw.), gelten für Banddateien dieselben Regeln wie für Plattendateien, allerdings ist ISAM-Verarbeitung von Banddateien nicht möglich.

9.3.2 Mehrbanddatei: Eine Datei auf mehreren Bändern

Eine „Mehrbanddatei“ ist eine Datei, die sich über mehrere Bänder erstreckt. Die Bänder werden unter dem Begriff „Bandmenge“ zusammengefasst. Mehrbanddateien werden also auf Bandmengen gespeichert. Für Dateien auf MBK besteht die Bandmenge aus mehreren Kassetten.

Aufbau einer Mehrbanddatei

Bei Mehrbanddateien mit Standardkennsätzen beginnt jedes Band mit dem VOL1-Kennsatz, anschließend folgen die HDR-Kennsätze und eine Abschnittsmarke. Auf dem ersten Band folgt der Abschnittsmarke der erste Abschnitt der Datei, abgeschlossen von einer weiteren Abschnittsmarke. An die Stelle der EOF-Kennsätze treten die EOV-Kennsätze, die das Bandende kennzeichnen und beim Bandwechsel vom System automatisch geschrieben werden.

Im HDR1-Kennsatz des zweiten Bandes hat das Feld „Dateiabschnittsnummer“ den Wert „0002“, da die HDR-Kennsätze den zweiten Dateiabschnitt einleiten. Entsprechend werden bei weiteren Bandwechseln EOV- und HDR-Kennsätze mit den jeweils aktuellen Dateiabschnittsnummern geschrieben.

Der letzte Dateiabschnitt wird schließlich mit EOF-Kennsätzen abgeschlossen.

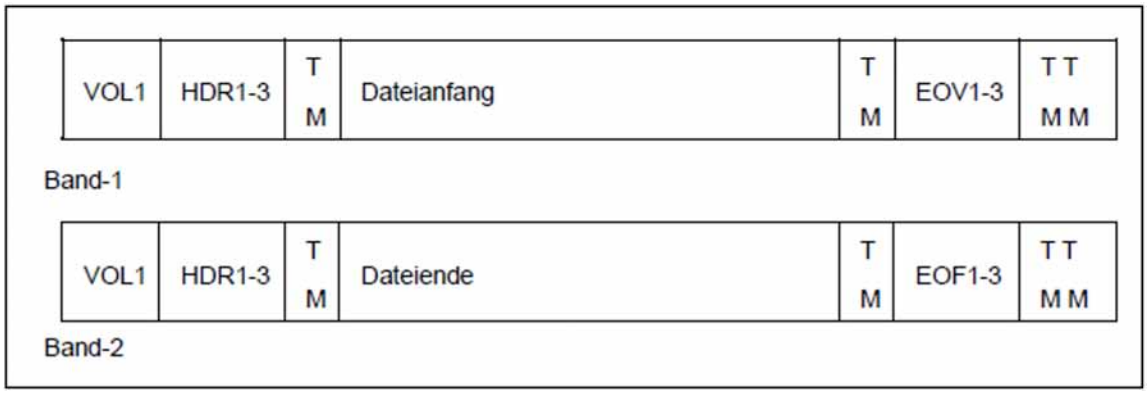


Bild 8: Mehrbanddatei

Mehrbanddatei erzeugen

Es kann bereits im Makro FILE bzw. im Kommando CREATE-FILE festgelegt werden, dass eine Datei erzeugt werden soll, die sich über mehrere Bänder erstrecken wird. Mit dem Operanden DEVICE bzw. DEVICE-TYPE wird festgelegt, ob die Datei auf Magnetband oder MBK geschrieben wird. Im VOLUME-Operanden kann dann angegeben werden, wie viele und welche Bänder voraussichtlich für die Datei benötigt werden.

Die Liste der angegebenen Archivnummern wird als Datenträgerliste in den Katalogeintrag übernommen.

Für bereits katalogisierte Eingabedateien kann der Benutzer im Kommando ADD-FILE-LINK, Operand PROCESS-VOLUME, eine Liste von Archivnummern angeben. Dadurch werden die im Katalog eingetragenen Datenträgerkennzeichen während der Verarbeitung ignoriert und nur die hier angegebenen Datenträger verwendet. Der Katalogeintrag wird jedoch nicht verändert.

Der Operand TAPE-SET-NAME im Kommando ADD-FILE-LINK verknüpft den TFT-Eintrag mit einer Datenträgerliste, die duch das Kommando CREATE-TAPE-SET in der „Tape Set Table“ (TST) erzeugt wurde. Diese TST wird bei der Verarbeitung aktualisiert und nach Abschluss der Verarbeitung für die Datenträgerliste des Katalogeintrags ausgewertet.

Umfasst die Volume-Liste mehrere Datenträger, kann mit dem Operanden MOUNT im Makro FILE bzw. mit dem Operanden PREMOUNT-LIST im Kommando CREATE-FILE bestimmt werden, wie viele Datenträger „sofort“ bereitgestellt werden müssen, d.h. wie viele Geräte belegt werden sollen. Fordert der Benutzer die Bereitstellung zweier Geräte, kann er diese während der Verarbeitung alternierend belegen und die Bandwechselzeiten stark verkürzen.

Sind im Benutzerprogramm keine Routinen zum Schreiben eigener Kennsätze vorgesehen, übernimmt das System das Schreiben aller Kennsätze sowie die Steuerung des Bandwechsels. Ob Standardkennsätze geschrieben werden, hängt vom Operanden LABEL in den Makros FILE oder FCB bzw. vom Operanden LABEL-TYPE im Kommando ADD-FILE-LINK ab. Wie der Bandwechsel abläuft, ist im Kapitel „EOV-Verarbeitung“ detailliert beschrieben.

9.3.3 Dateimenge: Mehrere Dateien auf einem Band

Werden mehrere Dateien auf einem Band gespeichert, bilden sie eine Dateimenge (= File Set); der Sonderfall „Dateimenge auf Bandmenge“ (= MF/MV-Set) ist in Abschnitt „MF/MV-Set: Dateimenge auf Bandmenge“ und Abschnitt „Verarbeitung von MF/MV-Sets“ beschrieben.

Aufbau einer Dateimenge

Ein Band mit Standardkennsätzen beginnt mit dem Bandanfangs-Kennsatz VOL1, dem die HDR-Kennsätze der ersten Datei folgen. In Abschnittsmarken eingeschlossen folgen die Daten dieser ersten Datei, anschließend ihre EOF-Kennsätze.

Da noch weitere Dateien auf dem Band folgen, schließt sich an die EOF-Kennsätze nur eine einfache Abschnittsmarke vor den HDR-Kennsätzen der zweiten Datei an, usw. Erst nach den EOF-Kennsätzen der letzten Datei steht die Doppel-Abschnittsmarke. In den HDR1- und EOF1-Kennsätzen wird in der „Dateifolgenummer“ (= File Sequence Number, FSEQ) die Position der Datei in der Dateimenge verankert.

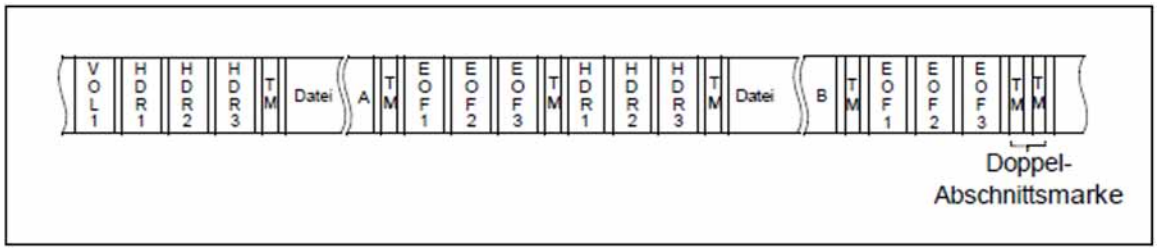


Bild 9: Dateimenge

Dateimenge erzeugen

Mit den Operanden DEVICE und VOLUME im Makro FILE bzw. den Operanden DEVICE-TYPE und VOLUME im Kommando CREATE-FILE wird festgelegt, auf welchem Datenträger die Datei gespeichert werden soll. Wird ein Band für mehrere Dateien verwendet, so überschreibt normalerweise jede Datei das Band ab Bandanfang, d.h., sie überschreibt eine evtl. vorher auf dem Band stehende Datei.

Soll eine Dateimenge entstehen, müssen die Dateien hintereinander auf das Band geschrieben werden. Dies wird in den Makros FILE und FCB über den Operanden FSEQ, im Kommando ADD-FILE-LINK über den Operanden FILE-SEQUENCE gesteuert. FSEQ bzw. FILE-SEQUENCE weisen einer Datei ihren Platz in der Dateimenge zu; die Nummerierung der Dateien über FSEQ bzw. FILE-SEQUENCE muss lückenlos aufsteigend sein. Auf diese Weise können auch Dateien mit gleichem Namen hintereinander auf ein Band geschrieben werden, sie sind durch den Operanden eindeutig identifizierbar.

Ist die gültige Nummer einer Datei unbekannt, kann der Benutzer im Makro mit FSEQ=UNK (= unknown) bzw. im Kommando mit FILE-SEQUENCE=UNKNOWN eine Eingabedatei suchen lassen oder mit FSEQ=NEW bzw. FILE-SEQUENCE=NEW bewirken, dass die Ausgabedatei an das Ende der Dateimenge geschrieben wird.

9.3.4 MF/MV-Set: Dateimenge auf Bandmenge

Ein Multifile/Multivolume-Set (= MF/MV-Set) ist das Ergebnis der Speicherung einer Dateimenge auf einer Bandmenge. Der Begriff MF/MV-Set bezieht sich nur auf den logischen Zusammenhang zwischen Dateimenge und Bandmenge. Das DVS unterhält keinen Daten-träger-Dateikatalog.

Aufbau eines MF/MV-Sets

Die erste Datei der Dateimenge beginnt auf dem ersten Band der Bandmenge.

Das Dateimengenkennzeichen ist im HDR1-Kennsatz einer jeden Datei enthalten; es ist die Archivnummer (VSN) des ersten Bandes, das den ersten oder einzigen Abschnitt der ersten Datei der Dateimenge enthält – das heißt: das Dateimengenkennzeichen aller Dateien einer Dateimenge ist die Archivnummer des ersten Bandes der zugehörigen Bandmenge.

Beispiel: MF/MV-Set

Folgende Dateien sind im Dateikatalog eingetragen:

```
Datei A        (FSEQ=1)
               Bandfolge: 000001, 000002
Datei B        (FSEQ=2)
               Bandfolge: 000001, 000002, 000003, 000004
Datei C        (FSEQ=3)
               Bandfolge: 000004, 000005
Datei D        (FSEQ=4)
               Bandfolge: 000005, 000006, 000007, 000008
```

Zum Zeitpunkt der Erstellung wurden die folgenden Bandmengen in den notwendigen CREATE-FILE bzw. ADD-FILE-LINK-Kommandos angegeben:

```
Datei A :  000001, 000002
Datei B :  000001, 000002, 000003, 000004
Datei C :  000003, 000004, 000005, 000006
Datei D :  000004, 000005, 000006, 000007, 000008
```

Die Dateimenge setzt sich aus (A,B,C,D) zusammen. Die zugehörige Bandmenge ist: (000001, 000002, 000003, 000004, 000005, 000006, 000007, 000008).

Erläuterung zum Bild

- (n,n) = (Dateifolgenummer, Dateiabschnitt)
- Die Datei C beginnt auf dem Band 000004 mit einem leeren Dateiabschnitt (siehe auch Bild 10 (Leere Dateiabschnitte)).
- Die Datei D endet auf Band 000008 mit einem leeren Dateiabschnitt.

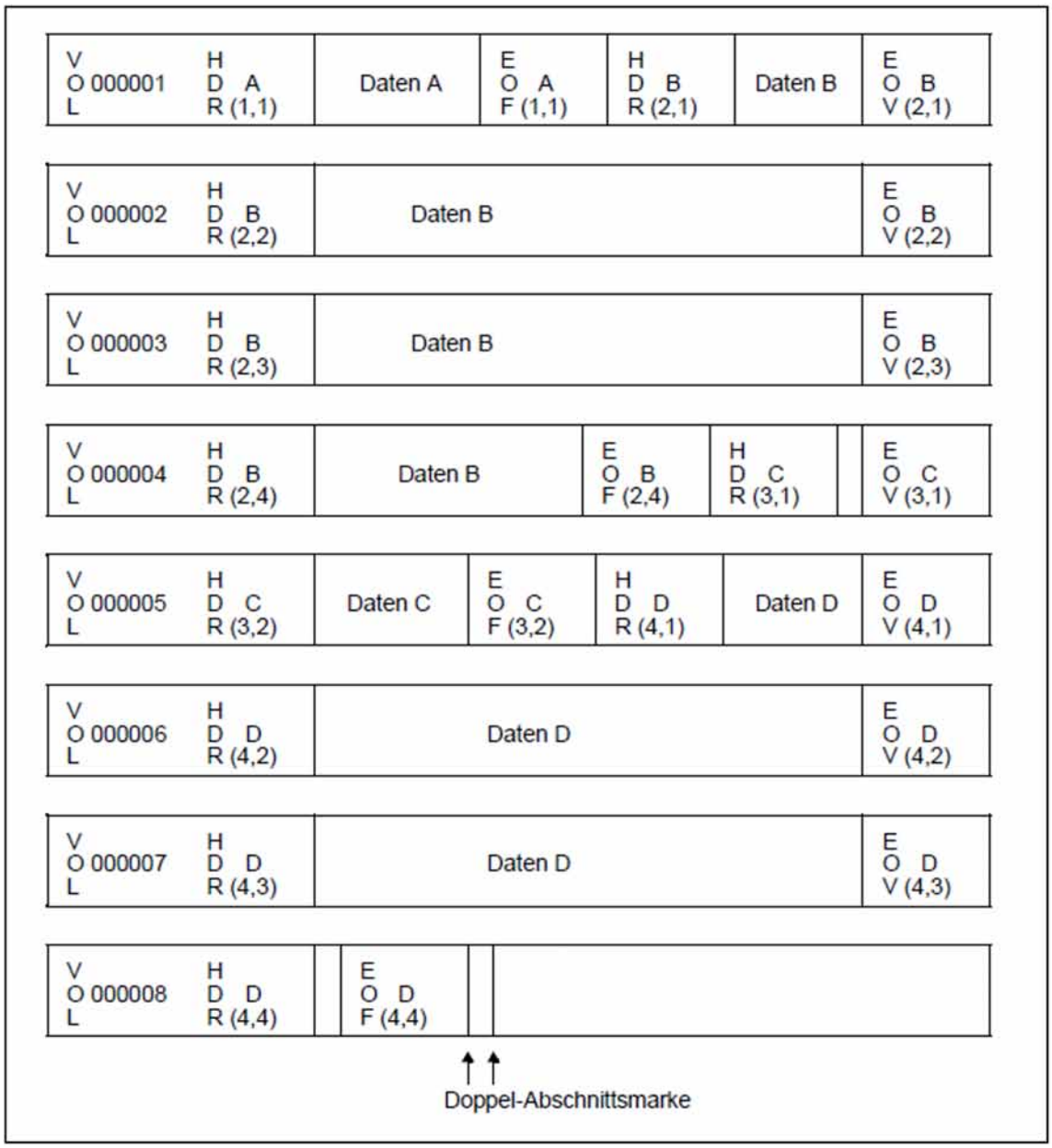


Bild 10: MF/MV-Set

MF/MV-Set erstellen

Ein MF/MV-Set entsteht wie eine Dateimenge, die sich allerdings über mehrere Bänder erstreckt. Im FILE-Aufruf müssen dazu die Operanden DEVICE, VOLUME und FSEQ versorgt werden, für jede Datei der Dateimenge ist ein FILE-Makro aufzurufen. Im Kommando

CREATE-FILE müssen die Operanden DEVICE-TYPE und VOLUME versorgt werden, im Kommando ADD-FILE-LINK der Operand FILE-SEQUENCE. Beide Kommandos sind für jede Datei der Dateimenge einzugeben.

Mit dem Operanden DEVICE im Makro bzw. DEVICE-TYPE im Kommando legt der Benutzer fest, mit welchem Datenträgertyp er arbeiten will (bzw. welchen Gerätetyp er dazu benötigt). Im VOLUME-Operanden werden die Archivnummern der Bänder aufgelistet, auf denen die Datei gespeichert werden soll. Der Operand FSEQ im Makro bzw. FILE-SEQUENCE im Kommando gibt die Position der Datei in der Dateimenge an. Beginnend mit FSEQ/FILE-SEQUENCE=1 dürfen keine Lücken in dieser Folge entstehen, und die Dateien müssen in der Reihenfolge dieser Nummern erstellt werden. Wird die Angabe FSEQ=NEW im Makro FILE bzw. die Angabe FILE-SEQUENCE=NEW im Kommando ADD-FILE-LINK verwendet, erstellt das BS2000 die Dateien jeweils am Ende der Dateimenge und weist ihnen die gültige Nummer zu. Gleichzeitig befreit es den Benutzer davon, in Prozeduren oder Programmen diesen Operanden variabel halten zu müssen.

In den Katalogeintrag der Datei werden übernommen

- die Bandfolge, das ist die Datenträgerliste der VOLUME-Operanden, ab dem Band, auf dem die Datei beginnt
- die FSEQ- bzw. FILE-SEQUENCE-Nummer, die der Benutzer in FILE/FCB bzw. ADD-FILE-LINK angegeben oder die das System ermittelt hat (bei FSEQ=NEW bzw. bei FILE-SEQUENCE=NEW)

Das System schreibt die Standardkennsätze automatisch (abhängig vom LABEL-Operanden des FILE-/FCB-Makroaufrufs bzw. des Kommandos ADD-FILE-LINK) und führt auch den Bandwechsel automatisch durch.

Wird im VOLUME-Operanden des Makros FILE bzw. des Kommandos CREATE-FILE eine Bandmenge angegeben, müssen diese Bänder frei sein.

Die Kennsatzstufe des ersten und aller weiteren Bänder der Bandmenge wird mit dem LABEL-Operanden in den Makros FILE und FCB bzw. im Kommando ADD-FILE-LINK bei der Erstellung der ersten Datei der Dateimenge festgelegt. Wird eine neue Datei hinzugefügt, muss sie mit Kennsätzen der gleichen Kennsatzstufe, wie im Bandkennsatz (VOL1) hinterlegt, erzeugt werden. Erstreckt sich eine Datei über mehrere Bänder, werden die Folgebänder mit der gleichen Kennsatzstufe erstellt.

9.3.5 Leere Dateiabschnitte

In MF/MV-Sets können als Sonderfälle leere Dateiabschnitte am Bandanfang oder Bandende entstehen, wenn z.B. beim Schreiben von Kennsätzen Bandende erkannt wird. Das DVS leitet dann Bandwechsel ein und beendet die Kennsatzverarbeitung auf dem Folgeband.

Leerer Dateiabschnitt am Bandanfang

Erkennt das System das Bandende, während es den letzten Datenblock einer Datei oder die Dateiende-Kennsätze schreibt, schließt es das Band mit Bandende-Kennsätzen und Doppel-Abschnittsmarke ab. Nach dem Bandwechsel schreibt es auf das Folgeband nach dem Bandanfangs-Kennsatz zunächst die Dateianfangs-Kennsätze der nicht abgeschlossenen Datei und anschließend nach einer Doppel-Abschnittsmarke die Dateiende-Kennsätze.

Ist die Datei die letzte Datei einer Dateimenge, wird als Abschluss die Doppel-Abschnittsmarke geschrieben. Folgen noch weitere Dateien, wird nur eine einfache Abschnittsmarke geschrieben und anschließend die HDR-Kennsätze der Folgedatei.

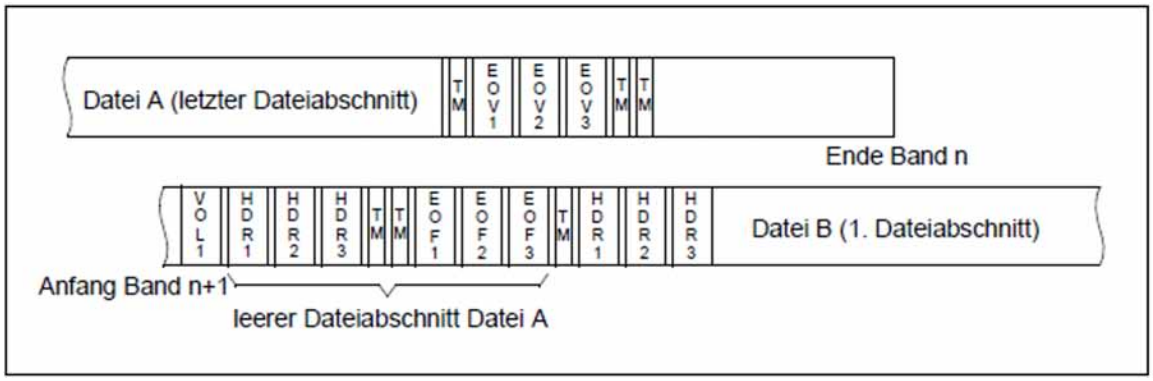


Bild 11: Leerer Dateiabschnitt am Bandanfang

Leerer Dateiabschnitt am Bandende

Erkennt das System beim Schreiben der Dateianfangs-Kennsätze die Bandendemarke, schreibt es anschließend sofort die Bandende-Kennsätze, eingeschlossen zwischen zwei Doppel-Abschnittsmarken.

Das Folgeband beginnt mit Band- und Dateianfangs-Kennsätzen, allerdings wird die Dateiabschnittsnummer im HDR1-Kennsatz um eins erhöht. Nach einer Abschnittsmarke folgt der erste Dateiabschnitt.

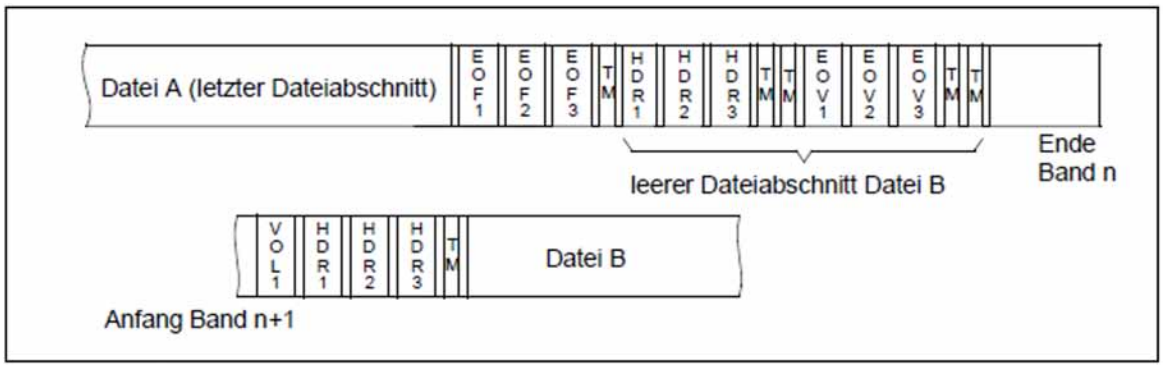


Bild 12: Leerer Dateiabschnitt am Bandende

9.4 Verarbeitung von Dateimengen

Dateimenge überschreiben

Ein Band kann nur dann überschrieben werden, wenn das Freigabedatum erreicht und der Schreibzugriff erlaubt ist.

Soll eine Datei überschrieben werden, positioniert das DVS entsprechend der Dateifolgenummer. Die Dateischutzmerkmale wie Kennwörter usw. werden nur bei katalogisierten Dateien berücksichtigt.

> ¡ Wird innerhalb einer Dateimenge eine Datei überschrieben oder erweitert, kann auf die Folgedateien in der Dateimenge nicht mehr zugegriffen werden! Der Rest des Bandes ist implizit gelöscht.

Ist die in den Makros FILE und FCB bzw. im Kommando ADD-FILE-LINK genannte Datei katalogisiert, muss die Dateifolgenummer im Operanden FSEQ bzw. FILE-SEQUENCE mit der des Katalogeintrags übereinstimmen. Ist die Datei noch nicht katalogisiert, positioniert das DVS entsprechend der FSEQ- bzw. FILE-SEQUENCE-Angabe; die Datei kann dann ohne Überprüfung des Dateinamens überschrieben werden.

Ist die Datei nicht katalogisiert und ist die FSEQ- bzw. FILE-SEQUENCE-Angabe um eins größer als die höchste Dateifolgenummer der Dateimenge, wird die Dateimenge um eine Datei erweitert.

Dateimenge erweitern

Soll eine Dateimenge um eine Datei erweitert werden, muss entweder im Makro FILE oder FCB der Operand FSEQ=NEW bzw. im Kommando ADD-FILE-LINK der Operand FILE-SEQUENCE=*NEW angegeben werden oder eine Zahl, die um 1 höher ist als die letzte zur Dateimenge gehörende Dateifolgenummer. Die Datei darf noch nicht katalogisiert sein.

Bei FSEQ=NEW bzw. FILE-SEQUENCE=*NEW positioniert das DVS auf das Ende der Dateimenge. Liegt dieses in der im Makro FILE bzw. im Kommando ADD-FILE-LINK angegebenen Bandmenge, wird die neue Datei mit einer Dateifolgenummer, die um 1 größer ist als die der bisher letzten Datei, auf das betreffende Band geschrieben.

Ist in FILE-/FCB-Makroaufruf SECLEV=OPR angegeben, muss der Schutzgrad der Datei kleiner oder gleich dem der vorausgehenden Datei sein. Der gleiche Sachverhalt trifft zu, wenn im Kommando ADD-FILE-LINK der Operand OVERWRITE-PROTECTION=*YES angegeben wurde.

Erweitern einer Dateimenge um mehrere Dateien (Programmschnittstelle)

Wenn Sie sicher sind, dass alle Dateien einer Dateimenge auf demselben Band Platz finden, können Sie die folgende Verarbeitungsfolge ausführen:

```
         FILE   AAA,VOLUME=000001,FSEQ=1,DEVICE=TAPE,LINK=OUTP1
         OPEN   OUT1,OUTPUT
         .
         .
         CLOSE  OUT1,LEAVE
         FILE   BBB,VOLUME=000001,FSEQ=NEW,DEVICE=TAPE,LINK=OUTP2
         OPEN   OUT2,OUTPUT
         .
         .
         CLOSE  OUT2,LEAVE
         FILE   CCC,VOLUME=000001,FSEQ=NEW,DEVICE=TAPE,LINK=OUTP3
         OPEN   OUT3,OUTPUT
         .
         .
         CLOSE  OUT3,LEAVE
         .
         .
OUT1     FCB   LINK=OUTP1...
OUT2     FCB   LINK=OUTP2...
OUT3     FCB   LINK=OUTP3...
```

Ist der Umfang einer Datei der Dateimenge (z.B. der ersten Datei) nicht abschätzbar, könnten Sie die Bandmengen in den FILE-Makroaufrufen für die Dateien AAA, BBB und CCC wie folgt erweitern:

`VOLUME=(000001,000002)`

Setzt sich jetzt die Datei AAA auf dem zweiten Band fort, fordert das DVS beim Erstellen der Dateien BBB und CCC jeweils erneut das erste Band an, um das Ende der Dateimenge festzustellen. Ist die Datei AAA so groß, dass Datei BBB auf dem zweiten Band keinen Platz mehr findet, fordert das DVS ein weiteres freies Band an. Anschließend führt jedoch die OPEN-Verarbeitung für die Datei CCC zum Fehler, weil die Dateimenge auf einem Band endet, das im FILE-Makroaufruf nicht angegeben wurde und daher nicht Teil der Bandmenge ist.

Sie können jedoch über den TSET-Operanden des FILE-Makros Datei- und Bandmenge temporär miteinander verknüpfen. Im TSET-Operanden geben Sie den Namen eines „Tape Sets“ an, das ist eine temporäre Datenträgerliste, die in der TST (Tape Set Table) geführt wird. Ein TST-Eintrag ist eine Tabelle, in die die Archivnummern der Bänder eingetragen werden, die – über einen TSET-Namen verknüpft – für einen Auftrag angefordert wurden.

Im folgenden Beispiel wird wieder die Dateimenge AAA, BBB und CCC erzeugt.

```
START
        LDBASE 10
        USING *,10
        .
         FILE   AAA,LINK=OUTP1,VOLUME=000001,DEVICE=TAPE,FSEQ=1,TSET=SET1
         OPEN   OUT1,OUTPUT
         .
         .
         CLOSE  OUT1,LEAVE
         FILE
BBB,LINK=OUTP2,DEVICE=TAPE,FSEQ=NEW,TSET=SET1,VOLUME=000002
         OPEN   OUT2,OUTPUT
         .
         .
         CLOSE  OUT2,LEAVE
         FILE   CCC,LINK=OUTP2,DEVICE=TAPE,FSEQ=NEW,TSET=SET1
         OPEN   OUT2,OUTPUT
         .
         .
         CLOSE  OUT2,LEAVE
         .
         .
OUT1     FCB    LINK=OUTP1...
OUT2     FCB    LINK=OUTP2...
OUT3     FCB    LINK=OUTP3...
```

Alle Dateien beziehen sich auf den gleichen TST-Eintrag, die Verknüpfung von Datei und TST-Eintrag erfolgt über die TFT (Task File Table), die mit dem LINK-Operanden für jede Datei angelegt wird. Das DVS organisiert automatisch die Anordnung der Dateimenge auf der gemeinsamen Bandmenge.

Der TST-Eintrag wird aufgebaut, wenn im FILE-Makroaufruf der LINK-Operand angegeben ist und gleichzeitig im TSET-Operanden ein neuer TST-Name angegeben wird. Bei einer Wiederholung der TSET-Angabe in weiteren FILE-Makroaufrufen für andere Dateien wird kein neuer TST-Eintrag angelegt. Der entsprechende TFT-Eintrag erhält lediglich einen Zeiger auf den TST-Eintrag, und im TST-Eintrag wird ein Dateizähler hochgezählt.

Bei Bandwechsel aktualisiert das DVS einen Zeiger im TST-Eintrag, der auf die Archivnummer des Bandes verweist, das das (bisherige) Ende der Dateimenge enthält. Die Bandmenge muss also nicht jedes Mal nach dem Ende der Dateimenge durchsucht werden, wenn eine neue Datei hinzugefügt werden soll, und unnötiger Bandwechsel wird vermieden.

> ¡ Arbeiten Sie mit einer TSET, müssen Sie beim Erstellen der ersten Datei einer Dateimenge FSEQ=1 angeben. Für weitere Dateien der Dateimenge dürfen im VOLUME-Operanden Archivnummern aufgelistet werden, die im TST-Eintrag nicht bekannt sind (siehe folgende Beispiele).
> In jedem Auftrag können mehrere TST-Einträge angelegt werden.

Beispiel 1

Ein bestimmtes Band als Erstes Band einer Bandmenge anfordern:

```
FILE AAA,LINK=OUTP,DEVICE=TAPE,VOLUME=000001,FSEQ=1,TSET=T
```

Beispiel 2

Ein beliebiges freies Band anfordern:

```
FILE AAA,LINK=OUTP,DEVICE=TAPE,FSEQ=1,TSET=T[,VOLUME=PRIVATE]
```

Beispiel 3

Eine Dateimenge wurde folgendermaßen eingerichtet:

```
FILE A,LINK=A,DEVICE=TAPE,VOLUME=(VOL1,VOL2),FSEQ=1,TSET=SET1
```

Die Dateimenge soll erweitert werden:

```
FILE C,LINK=C,DEVICE=TAPE,FSEQ=NEW,TSET=SET1,VOL=VOL3
```

Erweitern einer Dateimenge um mehrere Dateien (Kommandoschnittstelle)

Wenn Sie sicher sind, dass alle Dateien einer Dateimenge auf demselben Band Platz finden, können Sie die folgende kommando- und programminterne Verarbeitungsfolge ausführen:

Kommando	Programm
CREATE-FILE FILE-NAME=A,SUPPORT=*TAPE(VOLUME=000001, DEVICE-TYPE=xyz)	Ausgabedatei A anlegen
ADD-FILE-LINK LINK-NAME=A,FILE-NAME=A, SUPPORT=*TAPE(FILE-SEQUENCE=1)	Ausgabedatei A eröffnen : Datei A schließen
CREATE-FILE FILE-NAME=B,SUPPORT=*TAPE(VOLUME=000001,DEVICE-TYPE=xyz)	Ausgabedatei B anlegen
ADD-FILE-LINK LINK-NAME=B,FILE-NAME=B, SUPPORT=*TAPE(FILE-SEQUENCEC=*NEW)	Ausgabedatei B eröffnen : Datei B schließen
CREATE-FILE FILE-NAME=C,SUPPORT=*TAPE(VOLUME=000001, DEVICE-TYPE=xyz)	Ausgabedatei C anlegen
ADD-FILE-LINK LINK-NAME=C,FILE-NAME=C, SUPPORT=*TAPE(FILE-SEQUENCE=*NEW)	Ausgabedatei C eröffnen : Datei C schließen

Tabelle 27: Dateimenge erweitern (Beispiel 1)

Ist der Umfang einer Datei der Dateimenge nicht abschätzbar, könnten Sie die Bandmengen in den CREATE-FILE-Kommandos für die Dateien A, B und C wie folgt erweitern:

```
VOLUME=(000001,000002)
```

Setzt sich jetzt die Datei A auf dem zweiten Band fort, fordert das DVS beim Erstellen der Dateien B und C jeweils erneut das erste Band an, um das Ende der Dateimenge festzustellen. Ist die Datei A so groß, dass Datei B auf dem zweiten Band keinen Platz mehr findet, fordert das DVS ein weiteres freies Band an. Anschließend führt jedoch die OPEN-Verarbeitung für die Datei C zum Fehler, weil die Dateimenge auf einem Band endet, das im Kommando CREATE-FILE nicht angegeben wurde und daher nicht Teil der Bandmenge ist.

Sie können über den Operanden TAPE-SET-NAME im Kommando ADD-FILE-LINK Datei- und Bandmenge temporär miteinander verknüpfen. Im Operanden TAPE-SET-NAME geben Sie den Namen eines „Tape Sets“ an, das ist eine temporäre Datenträgerliste, die in der TST (Tape Set Table) geführt wird. Die TST ist eine Tabelle, in die die Archivnummern der Bänder eingetragen werden, die – über einen TSET-Namen verknüpft – für einen Auftrag angefordert wurden.

Im folgenden Beispiel wird wieder die Dateimenge A, B und C erzeugt.

Kommando	**Programm**
CREATE-TAPE-SET TAPE-SET-NAME=SET1, VOLUME=000001	Tapeset SET1 einrichten
CREATE-FILE FILE-NAME=A,SUPPORT=*TAPE(DEVICE-TYPE=xyz)	Ausgabedatei A anlegen
ADD-FILE-LINK LINK-NAME=A,FILE-NAME=A, SUPPORT=*TAPE(VOLUME-LIST=*TAPE-SET(TAPE-SET-NAME=SET1),FILE-SEQUENCE=1)	Ausgabedatei A eröffnen : Datei A schließen
CREATE-FILE FILE-NAME=B,SUPPORT=*TAPE(DEVICE-TYPE=xyz)	Ausgabedatei B einrichten
ADD-FILE-LINK LINK-NAME=B,FILE-NAME=B, SUPPORT=*TAPE(VOLUME-LIST=*TAPE-SET(TAPE-SET-NAME=SET1),FILE-SEQUENCE=*NEW)	Ausgabedatei B eröffnen : Datei B schließen
CREATE-FILE FILE-NAME=C,SUPPORT=*TAPE(DEVICE-TYPE=xyz)	Ausgabedatei C einrichten
ADD-FILE-LINK LINK-NAME=C,FILE-NAME=A, SUPPORT=*TAPE(VOLUME-LIST=*TAPE-SET(TAPE-SET-NAME=SET1),FILE-SEQUENCE=*NEW)	Ausgabedatei C eröffnen : Datei C schließen

Tabelle 28: Dateimenge erweitern (Beispiel 2)

Beim Schließen der Datei muss das Band jeweils auf Dateiende positioniert werden.

Alle Dateien beziehen sich auf den gleichen TST-Eintrag; die Verknüpfung von Datei und TST-Eintrag erfolgt über die TFT (Task File Table), die mit dem Operanden LINK-NAME für jede Datei angelegt wird. Das DVS organisiert automatisch die Anordnung der Dateimenge auf der gemeinsamen Bandmenge.

Der TST-Eintrag wird mit den Kommandos CREATE-TAPE-SET / EXTEND-TAPE-SET aufgebaut. Bei einer Wiederholung der Angabe TAPE-SET-NAME in ADD-FILE-LINK-Kommandos wird kein neuer TST-Eintrag angelegt. Der entsprechende TFT-Eintrag erhält lediglich einen Zeiger auf den TST-Eintrag, und in dem TST-Eintrag wird ein Dateizähler hochgezählt.

Bei Bandwechsel aktualisiert das DVS einen Zeiger in den TST-Eintrag, der auf die Archivnummer des Bandes verweist, das das (bisherige) Ende der Dateimenge enthält. Die Bandmenge muss also nicht jedes Mal nach dem Ende der Dateimenge durchsucht werden, wenn eine neue Datei hinzugefügt werden soll, und unnötiger Bandwechsel wird vermieden.

> ¡ Arbeiten Sie mit einer TSET, müssen Sie beim Erstellen der ersten Datei einer Dateimenge FILE-SEQUENCE=1 angeben. Für weitere Dateien der Dateimenge dürfen im Operanden VOLUME Archivnummern aufgelistet werden, die in dem TST-Eintrag nicht bekannt sind (s.u. Beispiel). In jedem Auftrag können mehrere TST-Einträge angelegt werden.

Beispiel: Dateimenge erzeugen

Mit den folgenden Kommandos fordern Sie ein bestimmtes Band als Erstes Band der Bandmenge an:

```
/CREATE-TAPE-SET TAPE-SET-NAME=T,VOLUME=000001
/CREATE-FILE FILE-NAME=A,SUPPORT=TAPE(DEVICE-TYPE=xyz)
/ADD-FILE-LINK LINK-NAME=A,FILE-NAME=A,SUPPORT=*TAPE(VOLUME-LIST=
 *TAPE-SET(TAPE-SET-NAME=T),FILE-SEQUENCE=1
```

Mit dem folgenden Kommando fordern Sie ein beliebiges freies Band an:

```
/CREATE-FILE FILE-NAME=A,SUPPORT=TAPE(VOLUME=*ANY,DEVICE-TYPE=xyz)
```

Die Dateimenge kann erweitert werden mit den folgenden Kommandos:

```
/CREATE-FILE FILE-NAME=B,SUPPORT=TAPE(DEVICE-TYPE=xyz)
/ADD-FILE-LINK LINK-NAME=B,FILE-NAME=B,SUPPORT=*TAPE(VOLUME-LIST=
 *TAPE-SET(TAPE-SET-NAME=T),FILE-SEQUENCE=*NEW
```

Beispiel: Dateimenge erweitern

Eine Dateimenge wurde wie folgt eingerichtet:

```
/CREATE-TAPE-SET TAPE-SET-NAME=SET1, VOLUME=(000001,000002)
/CREATE-FILE FILE-NAME=A,SUPPORT=TAPE(DEVICE-TYPE=xyz)
/ADD-FILE-LINK LINK-NAME=A,FILE-NAME=A,SUPPORT=*TAPE(VOLUME-LIST=TAPE-SET
 (TAPE-SET-NAME=SET1),FILE-SEQUENCE=1)
```

Die Dateimenge soll erweitert werden:

```
/CREATE-TAPE-SET TAPE-SET-NAME=SET2, VOLUME=000003
/CREATE-FILE FILE-NAME=C,SUPPORT=TAPE(DEVICE-TYPE=xyz)
/ADD-FILE-LINK LINK-NAME=C,FILE-NAME=C,SUPPORT=*TAPE(VOLUME-LIST=TAPE-SET
 (TAPE-SET-NAME=SET2),FILE-SEQUENCE=*NEW)
```

9.5 Verarbeitung von MF/MV-Sets

Dateizugriff im MF/MV-Set

Soll auf eine katalogisierte Datei innerhalb eines MF/MV-Sets zugegriffen werden, muss in den Makros FILE und FCB bzw. im Kommando ADD-FILE-LINK nur der Dateinamen angegeben werden. Das DVS entnimmt die zugehörigen Archivnummern und die Dateifolgenummer dem Katalogeintrag. Bei Eröffnung der Datei (OPEN) positioniert das DVS das erste Band der Datenträgerliste auf den Dateianfang, überprüft dann die Dateianfangskennsätze und erlaubt – falls zulässig – den Zugriff auf die Datei.
Der Ablauf von Kennsatzprüfung und Positionieren ist im Kapitel „Kennsatz-Verarbeitung beim Eröffnen von Banddateien“ detailliert beschrieben.

Soll auf eine bestehende, aber nicht katalogisierte Datei (= FOREIGN-Datei) zugegriffen werden, können bei Verwendung von Makros folgende Angaben gemacht werden:

- im FILE-Makroaufruf der Dateiname, der Operand STATE=FOREIGN, die Dateifolgenummer (FSEQ=) und im VOLUME-Operanden die Datenträgerliste
- Falls die Dateifolgenummer unbekannt ist, kann neben Dateinamen und VOLUME-Datenträgerliste der Operanden „FSEQ=UNK“ angegeben werden. Das DVS sucht die Datei auf der angegebenen Bandmenge, wobei es den Dateinamen als Suchargument verwendet. Dies setzt jedoch voraus, dass die FOREIGN-Datei mit Standardkennsätzen erstellt wurde

Bei Verwendung von Kommandos muss die Datei zunächst mit dem Kommando IMPORT-FILE importiert werden. Danach der Benutzer folgende Angaben machen:

- im Kommando ADD-FILE-LINK den Dateinamen und die Dateifolgenummer (Operand FILE-SEQUENCE=)
- Falls die Dateifolgenummer unbekannt ist, kann neben dem Dateinamen der Operand „FILE-SEQUENCE=UNKNOWN“ angegeben werden. Das DVS sucht die Datei auf der angegebenen Bandmenge, wobei es den Dateinamen als Suchargument verwendet. Dies setzt jedoch voraus, dass die FOREIGN-Datei mit Standardkennsätzen erstellt wurde

Enthält der FILE-Makroaufruf keine VSEQ-Angabe bzw. das Kommando ADD-FILE-LINK keine Angaben beim Operanden VOL-SEQUENCE-NUMBER, setzt das DVS voraus, dass der erste Abschnitt der Datei gesucht wird.

Zugriff in Richtung Band-/Dateianfang (REVERSE)

Beim Zugriff im REVERSE-Modus fordert das DVS den Datenträger an, der den letzten Dateiabschnitt enthält und positioniert an das Ende dieses Dateiabschnitts. Aus der im Katalogeintrag vermerkten Anzahl der Bänder, über die sich die Datei erstreckt, leitet das DVS ab, welcher Datenträger der Bandfolge das Dateiende enthält. Ist die Datei nicht katalogisiert (FOREIGN-Datei), setzt das DVS voraus, dass das letzte Band der Datenträgerliste das Dateiende enthält.

Wird im Makro FILE (Operand VSEQ) bzw. im Kommando ADD-FILE-LINK (Operand VOL-SEQUENCE-NUMBER) nur eine Bandfolgenummer angegeben, fordert das DVS das entsprechende Band an, da Dateiabschnitts- und Bandfolgenummer einander entsprechen. Es positioniert dann auf das Ende des Dateiabschnitts.

Zugriff zu einem beliebigen Dateiabschnitt

Soll eine Datei verarbeitet werden, die sich über mehrere Bänder erstreckt, kann die Datei über den Makro FILE (Operand VSEQ) bzw. über Kommando ADD-FILE-LINK (Operand VOL-SEQUENCE-NUMBER) auf einen beliebigen Dateiabschnitt positioniert werden. Das Benutzerprogramm kann dann diesen einen Dateiabschnitt oder ab diesem Dateiabschnitt alle weiteren bis zum Dateiende verarbeiten.

Soll das Programm nur einen Dateiabschnitt verarbeiten, müssen muss der VSEQ-Operanden folgendermaßen angegeben werden:

```
VSEQ=(L=(n))
```

Soll das Programm ab einem Dateiabschnitt alle weiteren bis zum Dateiende verarbeiten, muss der VSEQ-Operand folgendermaßen angegeben werden:

```
VSEQ=n
```

„n“ verweist jeweils auf das Band, mit dem ersten zu verarbeitenden Dateiabschnitt.

Wahlfreier Zugriff zu Dateiabschnitten

Dateiabschnitte einer Datei können in beliebiger Reihenfolge verarbeitet werden. Dazu ist im VSEQ- bzw. VOL-SEQUENCE-NUMBER-Operanden eine Liste von Dateiabschnittsnummern anzugeben. Die Einträge in der Liste beziehen sich jeweils auf ein Band der Bandfolge, auf der die Datei gespeichert ist.

Beispiel 1 (Programm-Schnittstelle)

Datei A erstreckt sich über die Bänder 00000A, 00000B, 00000C, 00000D, 00000E. Wird vom Benutzer der VSEQ-Operanden angegeben:

```
VSEQ=(L=(4,1,3))
```

und verarbeitet as Programm jeden angegebenen Dateiabschnitt vollständig, fordert das DVS nacheinander das Montieren der Bänder 00000D, 00000A, 00000C an.

Wahlweiser Zugriff zu Dateiabschnitten ist insbesondere dann von Bedeutung, wenn Dateiabschnitte einer Mehrbanddatei (Multi-Volume-File) zerstört oder nicht mehr lesbar sind und Sie die noch lesbaren Teile Ihrer Daten verarbeiten möchten.

Beispiel 2 (Programm-Schnittstelle)

Wäre im oben gezeigten Beispiel der zweite Dateiabschnitt der Datei A zerstört, könnten mit der folgenden Angabe die angegebenen Dateiabschnitte noch verarbeitet werden:

```
VSEQ=(L=(1,3,4,5)
```

Beispiel 3: Datei abschnittsweise verarbeiten (Kommando-Schnittstelle)

Datei A erstreckt sich über die Bänder 00000A, 00000B, 00000C, 00000D, 00000E. Wenn folgendes ADD-FILE-LINK-Kommando abgegeben wird:

```
/ADD-FILE-LINK FILE-NAME=A,
     SUPPORT=*TAPE(VOLUME-LIST=*CATALOG(VOL-SEQUENCE-NUMBER=(4,1,3))
```

und das Programm jeden angegebenen Dateiabschnitt vollständig verarbeitet, fordert das DVS nacheinander das Montieren der Bänder 00000D, 00000A, 00000C an.

Beispiel 4: Zugriff auf Dateiabschnitte (Kommando-Schnittstelle)

Wäre im oben gezeigten Beispiel der zweite Dateiabschnitt der Datei A zerstört, könnten mit folgendem Kommando die übrigen Dateiabschnitte noch verarbeitet werden:

```
/ADD-FILE-LINK FILE-NAME=A,
     SUPPORT=*TAPE(VOLUME-LIST=*CATALOG(VOL-SEQUENCE-NUMBER=(1,3,4,5))
```

Erweiterung einer Datei auf einem MF/MV-Set

Für die Dateierweiterung im MF/MV-Set gilt das Gleiche wie für die Dateierweiterung in Dateimengen:

> ¡ Bei der Erweiterung von Dateien im MF/MV-Set oder bei der Erweiterung von Dateiabschnitten werden nachfolgende Dateien oder Abschnitte implizit überschrieben und die Dateimenge zerstört!

Die Katalogeinträge der überschriebenen Dateien werden nicht automatisch gelöscht, verweisen also auf nicht mehr vorhandene Dateien. Jeder Versuch, auf eine implizit überschriebene Datei zuzugreifen, führt zu einem OPEN-Fehler. Der Benutzer ist für das Löschen der Katalogeinträge selbst verantwortlich.

Es kann vorkommen, dass durch implizites Überschreiben der n-te Dateiabschnitt einer Mehrbanddatei verloren geht. Der Benutzer kann jedoch z.B. durch die Angabe
VSEQ=n+1 im FILE-Makroaufruf bzw. durch die Angabe START-POSITION=n+1 im Kommando ADD-FILE-LINK nach wie vor auf den folgenden Abschnitt seiner Datei zugreifen.

Alle Dateien einer Dateimenge, die vor der überschriebenen/erweiterten Datei auf dem Band gespeichert sind, sowie Dateien, die auf Bändern einer Bandmenge vor bzw. nach dem aktuellen Band gespeichert sind, sind weiter verfügbar.

Die überschriebene/erweiterte Datei wird zur letzten Datei der Dateimenge.

Die ursprüngliche Bandmenge existiert nicht mehr. Die Bandmenge einschließlich des Bandes, auf dem eine Datei überschrieben/erweitert wurde, bildet jetzt eine neue Bandmenge. Jeder Versuch, von dieser neuen Bandmenge auf ein Band der „Restmenge“ zu positionieren, führt zu unvorhersehbaren Ergebnissen.

Implizites Überschreiben einer Datei wird verhindert, wenn die vorausgehende Datei einen höheren Schutzgrad (Sperrfrist, Zugriffsart) hat. Bei Angabe des SECLEV-Operanden mit dem Zusatz OPR in den Makros FILE und FCB bzw. des Operanden OVERWRITE-PROTECTION im Kommando ADD-FILE-LINK kann eine Datei nicht erstellt werden, wenn ihre Vorgängerdatei einen geringeren Schutzgrad hat.

Positionieren von MF/MV-Sets

Der Zugriff zu einer Datei in einem MF/MV-Set setzt das Positionieren der Bandmenge auf die gewünschte Datei voraus. Dieses Positionieren wird für katalogisierte Dateien mit Standardkennsätzen oder ohne Kennsätze automatisch vorgenommen, wenn die Datei eröffnet werden soll. Bei nicht-katalogisierten Dateien und Dateien, die mit Nicht-Standardkennsätzen erstellt wurden, ist der Benutzer für die korrekte Positionierung verantwortlich.

Das Positionieren erfolgt auf der Basis der Dateifolgenummer, die durch FSEQ bzw. FILE-SEQUENCE festgelegt wird. Jeder Datei auf einem MF/MV-Set wird beim Erstellen eine fortlaufende Dateifolgenummer zugeordnet. Wird eine Datei auf einem Folgeband fortgesetzt, bleibt ihre Dateifolgenummer gleich. Erst die Datei, die ihr auf dem Folgeband folgt, erhält eine um 1 größere Dateifolgenummer. Die Dateifolgenummer einer Datei wird im HDR1-Kennsatz und im Katalogeintrag festgehalten.

Will ein Benutzer auf eine katalogisierte Datei zugreifen, übernimmt das DVS die Dateifolgenummer und die Archivnummern der Bänder, auf denen die Datei gespeichert ist, aus dem Datei-Katalogeintrag.

Für den Zugriff zu einer nichtkatalogisierten Datei oder beim Erstellen einer neuen Datei, muss der Operand FSEQ in den Makros FILE und FCB bzw. der Operanden FILE-SEQUENCE im Kommando ADD-FILE-LINK angegeben werden. Andernfalls unterstellt das DVS, dass auf die erste Datei einer Dateimenge zugegriffen werden soll.

Wenn eine Datei auf dem Band eröffnet wird, wird die aktuelle Dateifolgenummer dem Aufruf der Makros FILE oder FCB bzw. des Kommandos ADD-FILE-LINK oder dem Katalogeintrag entnommen.

Das Band wird solange weiter positioniert, bis

- die aktuelle Dateifolgenummer mit der geforderten übereinstimmt
- in Zusammenhang mit FSEQ=NEW bzw. FILE-SEQUENCE=NEW das Ende des MF/MV-Sets erkannt wird: eine Doppel-Abschnittsmarke, die einer EOF-Kennsatzgruppe folgt; das Band wird um eine Abschnittsmarke zurückgesetzt.
- in Zusammenhang mit FSEQ=UNK bzw. FILE-SEQUENCE=UNKNOWN der Dateiname erkannt wird

Falls nötig, führt das DVS automatisch Bandwechsel durch, inklusive Kennsatzprüfungen (siehe Kapitel „Kennsatz-Verarbeitung beim Eröffnen von Banddateien“).

Wenn eine Datei mit OPEN REVERSE eröffnet werden soll, verläuft die Suche wie oben beschrieben, allerdings wird jeweils auf das Dateiende positioniert.

Für das Positionieren von Bändern mit Nicht-Standardkennsätzen muss der Benutzer in spezifischen EXLST-Routinen Sorge tragen.

9.6 Verarbeitung von Nicht-EBCDIC-Bändern

Magnetbänder können in verschiedenen Codes bearbeitet (gelesen bzw. beschrieben) werden. Daten und Kennsätze der Banddateien werden im jeweils angegebenen Code bearbeitet. Der Code ist ein Dateiattribut. Die Kennsatzprüfung erfolgt im angegebenen Code.

Mit folgenden Operanden werden bei SAM/BTAM die Codes im FILE- oder im FCB-Makroaufruf bzw. im Kommando ADD-FILE-LINK festgelegt:

CODE = EBCDIC	keine Umsetzung
CODE = ISO7	ISO-7-Bit-Code (128 Textzeichen)
CODE = OWN	alle übrigen Codes

Im EBCDI-Code und im ISO7-Code werden der nationale und internationale Zeichensatz mit den gleichen Bitkombinationen verschlüsselt (Doppelbelegung).

Fehlt der CODE-Operand im FILE- oder FCB-Makroaufruf bzw. im Kommando ADD-FILE-LINK, so wird die Angabe dem Katalogeintrag der Banddatei oder dem Kennsatz des Bandes entnommen.

Grundsätzlich werden die Codes EBCDIC oder ISO7 erkannt.
Kennsätze im Falle CODE=OWN werden nur dann erkannt, wenn im Benutzerprogramm eine Übersetzungstabelle angegeben ist. Für Lesen und Schreiben ist je eine Übersetzungstabelle (256 Byte) anzugeben. Die Adressen der Übersetzungstabellen für Lesen und Schreiben werden im FCB mit den Operanden TRTADR (Translate Table Address Read) und TRTADW (Translate Table Address Write) angegeben.

Der Benutzer kann festlegen, dass Banddateien mit einem beliebigen Code direkt (wie im EBCDIC ohne Umsetzung) gelesen werden.

Ist der Code einmal festgelegt, braucht sich der Benutzer um keine weiteren Einzelheiten der Codeumsetzung mehr zu kümmern. Er kann annehmen, dass der Datenverkehr mit der Banddatei wie im EBCDI-Code erfolgt.

9.7 Verwendung von MBK-Systemen

Vom BS2000 werden Magnetbandkassettensysteme (MBK-Systeme) mit unterschiedlichen Datenformaten (Anzahl der Spuren) und Aufzeichnungsverfahren (komprimiert oder unkomprimiert) unterstützt. Die in dieser Version unterstützten Typen finden Sie in der Freigabemitteilung von BS2000 OSD/BC.

Mit dem Volumetyp wird das Datenformat (also das HSI) bestimmt und damit implizit das MBK-Gerät ausgewählt. Beim Gerätetyp 3590E hat der Benutzer außerdem die Möglichkeit, das Aufzeichnungsverfahren über den Volumetyp zu bestimmen. Die Volumetyp-Tabelle finden Sie im Handbuch „Systeminstallation“ [15].

Da es einsichtig ist, dass die auf den verschiedenen MBK-Geräten erzeugten Dateien nicht so ohne weiteres auf anderen als auf den Geräten, auf denen sie erzeugt wurden, auch wieder gelesen und weiter verarbeitet werden können, muss sich der Systembetreuer an einige Vorschriften und Empfehlungen halten, die im Folgenden dargelegt werden.

Zugrunde liegende systemseitige Voraussetzungen

„Komprimierung“ als Volume-Eigenschaft

Ebenso wie das Datenformat wird das „Komprimieren“ als eine Volume-Eigenschaft betrachtet, d.h. alle Dateien einer Magnetbandkassette werden entweder komprimiert oder nicht-komprimiert aufgezeichnet.
In diesem Zusammenhang ist zu beachten, dass generell alle Kennsätze nicht-komprimiert geschrieben werden, also auch die von Dateien mit komprimierten Daten.

Datenformat und Aufzeichnungsverfahren als Kunden-Wunsch

Das System stellt nicht von sich aus ein Datenformat und ein Aufzeichnungsverfahren ein. Der Systembetreuer muss dem System den gewünschten Wert vorgeben (über eine Parameterangabe im Makro FILE mit dem Operanden DEV bzw. im Kommando CREATE-FILE oder IMPORT-FILE mit dem Operanden DEVICE-TYPE).

Aus systeminternen Gründen kommt als Parameter für diese Angaben im Makro FILE bzw. im Kommando CREATE-FILE nur der bereits bestehende DEV- bzw. DEVICE-TYPE-Parameter infrage. Dies hat zur Folge, dass damit sowohl das zu verwendende Gerät als auch das Aufzeichnungsverfahren bestimmt wird. Die ganze Verantwortung bzgl. der Verwendung des Parameters liegt also bei Ihnen. Gegebenenfalls müssen Sie in bisher schon bestehenden Prozeduren diesen Parameter ändern.

Einen übergeordneten Begriff für „MBK“ (ohne Festlegung weiterer Optionen) gibt es nicht.

Sie müssen also der Angabe des DEV- bzw. DEVICE-TYPE-Parameters besondere Beachtung schenken, um die gewünschte Behandlung Ihrer Dateien zu gewährleisten.

Welche Angaben müssen Sie also bzgl. des DEV-Wertes machen?

Hier sind nun verschiedene Fälle zu unterscheiden:

Neu-Erstellen einer Datei als Erste einer Multifile-Konfiguration

Durch Ihre DEV- bzw. DEVICE-TYPE-Angabe wird das zu verwendende Gerät und ebenso das Aufzeichnungsverfahren bestimmt.

Bearbeiten (Lesen) einer bestehenden Datei

Beim Lesen einer Datei müssen zwei Fälle unterschieden werden:

1. Es existiert ein Katalogeintrag. Dann ist dort der Volumetyp hinterlegt. Vom System wird ein passendes Gerät ausgewählt und das Lesen der Daten geschieht im richtigen Modus.

2. Ein Katalog-Eintrag existiert nicht, die Datei wird als „FOREIGN“ behandelt. In diesem Fall liegt es in Ihrer Verantwortung, beim Absetzen des DEV- bzw. des DEVICE-TYPE-Parameters das richtige Gerät auszuwählen, d.h. Sie müssen wissen, in welcher Form die Daten Ihrer Magnetbandkassette aufgezeichnet wurden.

Für STD-Label-behaftete Dateien übernimmt das DVS die Überprüfung der Verträglichkeit von Angaben in FILE oder IMPORT-FILE bzw. von Katalogangaben mit den Angaben, die in den Kennsätzen von bestehenden Dateien hinterlegt sind. Die Kennzeichnung einer Datei (und damit des ganzen Volumes) als „komprimiert“ oder nicht ist im HDR2 (und im EOF2)-Label in den für BS2000 reservierten Bytes 34 und 35 hinterlegt.

Im Falle einer Diskrepanz bei den von Ihnen gemachten Angaben bzw. den Katalogangaben mit den im Kennsatz hinterlegten Informationen erfolgt eine Abweisung des Auftrags vom System.

Fortscheiben von Dateien und Erzeugen von Folge-Dateien

Beim Fortschreiben einer bereits teilweise bestehenden Datei mit OPEN EXTEND oder IN-OUT wird durch Prüfung nach Übereinstimmung des Kennzeichens aus dem Datei-Kennsatz mit dem Verarbeitungswunsch gewährleistet, dass die Datei im gleichen Komprimierungsmodus weitergeschrieben wird.

Im Falle des Erstellens einer Folgedatei auf einem Multifile-Band bestimmt die erste auf dem Band befindliche Datei den allein wählbaren Modus. Das Kennzeichen im Kennsatz der ersten Datei wird dazu verwendet, um die Forderung „Komprimierung als Volume-Eigenschaft“ zu gewährleisten.

Im Falle von Nicht-Verträglichkeiten erfolgt eine Abweisung des Auftrags durch das System.

> **Hinweis**
>
> Alle vom DVS durchgeführten Überprüfungen können sich nur auf Bänder beziehen, die einerseits im STD-Label-Format erstellt sind, und andererseits das „BS2000“-Kennzeichen im HDR2-Kennsatz beinhalten. Da das Komprimierungs-Kennzeichen im HDR2 nicht in einem genormten Feld der Kennsätze liegt, ist bei Bändern fremder Hersteller (oder auch bei Bändern im NSTD-/NO-Format) keine eindeutige Identifizierung des Komprimierungszustandes möglich. Beim Bearbeiten von Bändern, die keine Standard-Kennsätze haben, liegt die Verantwortung für den „richtigen“ Komprimierungsmodus bei Ihnen.

10 Dateigenerationsgruppen (FGG)

Dateien, die in chronologischem Zusammenhang stehen und gleiche Dateieigenschaften besitzen, können im BS2000 in sog. „Dateigenerationsgruppen“ (File Generation Group, FGG) verwaltet werden. Die zu einer FGG gehörenden Dateien werden als „Dateigenerationen“ oder kurz als „Generationen“ bezeichnet.

Dateigenerationsgruppen sind konzipiert für mehr oder minder langfristiges Speichern von großen Datenmengen, z. B. von Daten aus regelmäßig wiederkehrenden Verarbeitungen oder für den Aufbau von Magnetbandpools, in denen z.B. Buchhaltungsdaten gespeichert werden. Sie bieten so gleichzeitig eine komfortable Methode der Datensicherung nach dem Großvater-Vater-Sohn-Prinzip.

Nicht in Dateigenerationsgruppen gespeichert werden können ablauffähige Programme oder Prozeduren. Werden Programme/Prozeduren als Dateigenerationen gespeichert, müssen sie in eine „normale“ Datei kopiert werden, bevor sie zum Ablauf gebracht werden können.

Der Aufbau des Gruppennamens folgt den Regeln zur Bildung von Dateinamen im BS2000 (siehe Abschnitt „Dateinamen“), mit der Einschränkung, dass die maximale Länge nur 34 Zeichen beträgt, da für die Identifizierung der Generationen eine Generationsnummer im Dateinamen verankert wird. Dateigenerationen sind also unter dem Gruppennamen mit Zusatz ihrer Generationsnummer ansprechbar.

```
Gruppenname:      [:catid:][$userid.]dateiname

Generationsname:  [:catid:][$userid.]dateiname(generationsnr)
```

10.1 Generationsnummer

In den Generationsnummern spiegelt sich der chronologische Zusammenhang zwischen den Generationen wider, da die Generationsnummern „laufende Nummern“ bei der Erstellung der Generationen sind.

Im Katalogeintrag der Dateigeneration ist immer die sog. „absolute Generationsnummer“ vermerkt. Diese Generationsnummer wird beim Einrichten neuer Generationen jeweils um 1 hochgezählt. Obere Grenze ist die Zahl 9999, die folgende Generation wird dann mit der Generationsnummer 0001 katalogisiert. Das Zeichen „*“ vor der Generationsnummer zeigt an, dass die absolute Generationsnummer angegeben wird; führende Nullen können weggelassen werden.

```
absolute Generationsnummer: (*zahl); 0001 <= zahl <= 9999
```

Damit Programme oder Prozeduren, in denen Dateigenerationen verarbeitet oder erstellt werden, nicht für jede Verarbeitung mit der neuen absoluten Generationsnummer aktualisiert werden müssen, wird im Katalogeintrag der Dateigenerationsgruppe eine Basis für die Verwendung relativer Generationsnummern definiert.
Der Benutzer kann eine beliebige – existente – Generation als Bezugspunkt für die relative Indizierung wählen und diesen Bezugspunkt mit dem Makro CATAL, Operand BASE, bzw. mit den Kommandos CREATE-FILE-GROUP und MODIFY-FILE-GROUP-ATTRIBUTES, Operand BASE-NUMBER, definieren. Welche (absolute) Generationsnummer aktueller Bezugspunkt ist, ist dem Feld „BASE-NUM“ des Gruppeneintrags zu entnehmen.
Das DVS erkennt relative Generationsnummern an den Vorzeichen „+“ oder „-“ an Stelle des „*“ der absoluten Generationsnummer.

```
relative Generationsnummer: (+zahl) oder (-zahl); 0 <= zahl <= 99
```

Beispiel: Zuordnung relative / absolute Generationsnummern

Eine Dateigenerationsgruppe GROUP soll aus den katalogisierten Generationen 20 bis 23 bestehen; der Katalogeintrag enthält die Information BASE-NUM=21. Es ergibt sich folgende Zuordnung von relativen zu absoluten Generationsnummern:

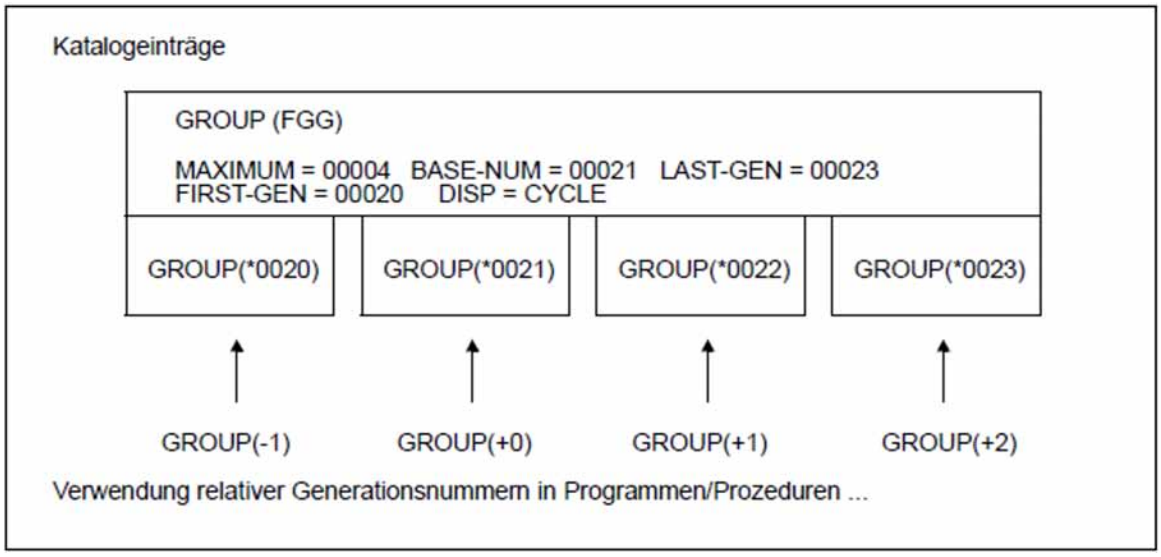


Bild 13: Absolute/relative Generationsnummern

10.2 Dateigenerationsgruppe erstellen

Werden Dateigenerationsgruppen/Dateigenerationen katalogisiert – sei es beim Erstellen oder beim Importieren – ist darauf zu achten, dass die FGG immer vollständig im entsprechenden Pubset katalogisiert sein muss. Das heißt, es darf in der katalogisierten Generationenfolge keine Lücke entstehen. Dateigenerationsgruppen können auch auf mehreren Pubsets gleichzeitig katalogisiert sein, wenn sie z.B. in mehrere Pubsets gleichzeitig importiert werden – sie müssen dann aber in jedem Pubset vollständig katalogisiert sein.

Dateiengenerationen und Dateigenerationsgruppen können auch auf einem Snapset gesichert werden. Einzelne Dateigenerationen können nur mit der gesamten Dateigenerationsgruppe von einem Snapset restauriert werden.

10.2.1 Gruppeneintrag erstellen

Dateigenerationsgruppen werden über ihren Gruppeneintrag verwaltet. Bevor Dateigenerationen erstellt werden können, muss zunächst dieser Gruppeneintrag eingerichtet werden.

Der Gruppeneintrag für eine Dateigenerationsgruppe kann mit dem Makro CATAL bzw. mit dem Kommando CREATE-FILE-GROUP erstellt werden. Für die Eigenschaften, die beim Erstellen nicht über die entsprechenden Operanden im Makro- oder Kommandoaufruf versorgt werden, übernimmt das DVS eine Voreinstellung in den Katalogeintrag. Der Benutzer kann die im Gruppeneintrag festgelegten Eigenschaften – zumindest für Plattendateien – später jederzeit ändern.

Der Operand DISP des Makros CATAL bzw. der Operand OVERFLOW-OPTION der Kommandos CREATE-FILE-GROUP und MODIFY-FILE-GROUP-ATTRIBUTES legt fest, wie das DVS beim Erreichen der Maximalzahl gleichzeitig katalogisierter Generationen mit den dann „überzähligen“ ältesten Generationen verfährt. So kann z.B. bestimmt werden, dass die Datenträger der überzähligen Generationen für die Speicherung der neuen Generationen verwendet werden. Bei Dateien auf Privatplatten wird zuerst die neue Generation erstellt, bevor die alte Generation gelöscht wird. Bei Speicherplatzmangel kann dieses Verhalten dazu führen, dass die neue Generation nicht angelegt werden kann, wenn (noch) nicht genügend Speicherplatz frei ist.

Standardmäßig wird jeweils der Katalogeintrag der ältesten Generation gelöscht.

Über den Gruppeneintrag werden im Makro- und Kommandoaufruf für alle Generationen der Dateigenerationsgruppe die Dateisicherungs- und Dateischutzmerkmale festgelegt. Die Schutzattribute BACL und GUARDS schützen sowohl die FGG, als auch jede einzelne Dateigeneration, mit der Ausnahme von Bandgenerationen. In einer mit diesen Attributen schreibgeschützten FGG kann ein Aufrufer ohne Schreibrecht keine neuen Dateigenerationen anlegen.

Auch die Verschlüsselungsmerkmale bei Dateiverschlüsselung werden über den Gruppeneintrag einheitlich für alle Generationen einer Gruppe festgelegt. Dabei sind die Bandgenerationen stets von einer Dateiverschlüsselung ausgenommen. Nur die ganze Gruppe, nicht eine einzelne Generation kann mit ENCRYPT-FILE (siehe "Voraussetzungen und Einschränkungen für Dateiverschlüsselung") in verschlüsselte Dateien umgewandelt werden. Neue Generationen (ausgenommen Bandgenerationen) erhalten die Verschlüsselungsmerkmale der Gruppe.

Merkmal	**Operand im Makro CATAL**	**Operand in den Kommandos CREATE-FILE-GROUP und MODIFY-FILE-GROUP-ATTRIBUTES**
Mehrbenutzbarkeit	SHARE=NO/YES /SPECIAL BASACL, GUARDS	USER-ACCESS=*OWNER-ONLY/ *ALL-USERS/ *SPECIAL BASIC-ACL, GUARDS
Zulässigkeit von Schreibzugriff	ACCESS=WRITE /READ	ACCESS=*WRITE/*READ
Kennwörter	RDPASS, WRPASS	READ-PASSWORD, WRITE-PASSWORD
Häufigkeit und Umfang der autom. Dateisicherung	BACKUP, LARGE	BACKUP-CLASS, SAVED-PAGES

autom. Datenzerstörung bei Speicherplatzfreigabe	DESTROY=NO/YES	DESTROY-BY-DELETE
Überwachung der DVS-Zugriffe	AUDIT	AUDIT

Tabelle 29: Merkmale für Dateigenerationen festlegen

Weitere Felder des Gruppeneintrags beziehen sich auf die Dateigenerationsgruppe als Ganzes:

- Maximalzahl gleichzeitig katalogisierter Generationen (MAXIMUM)
- Bezugswert für relative Generationsnummern (BASE-NUM)
- absolute Generationsnummer der jüngsten und ältesten katalogisierten Generation (LAST-GEN, FIRST-GEN)

Die übrigen Felder des Gruppeneintrags beziehen sich nur auf Zugriffe auf diesen Katalogeintrag (ACC-COUNT, CRE-DATE, EXPIR-DATE, CHANG-DATE, VERSION).

Die Katalogeinträge der Dateigenerationen sind also in Bezug auf die Dateisicherungsmerkmale mit dem Gruppeneintrag identisch. In den übrigen Eigenschaften können sie voneinander abweichen (HIGH-US-PA, CRE-DATE usw.).

Beispiel: Gruppeneintrag und Katalogeinträge von Dateigenerationen

`/show-file-attr max.group.2,information=all,select=by-attr(generation=yes)`

```
0000000000 :2OS6:$ULR.MAX.GROUP.2 (FGG)
   ------------------------------ HISTORY      ------------------------------
   CRE-DATE   = 2014-10-12  ACC-DATE   = NONE        CHANG-DATE = NONE
   CRE-TIME   =   14:08:14  ACC-TIME   = NONE        CHANG-TIME = NONE
   ACC-COUNT  = 0           S-ALLO-NUM = 0
   ------------------------------ SECURITY     ------------------------------
   READ-PASS  = YES         WRITE-PASS = NONE        EXEC-PASS  = NONE
   USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
   AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2014-10-12
   DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
   SP-REL-LOCK= NO          ENCRYPTION = *NONE
   ------------------------------ BACKUP       ------------------------------
   BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 0
   MIGRATE    = ALLOWED
   #BACK-VERS = 0
 ------------------------------ GENERATION-INFO -------------------------
   MAXIMUM    = 3           BASE-NUM   = 11          OVERFL-OPT = CYCL-REPL
   FIRST-GEN  = 11          LAST-GEN   = 13
 0000000024 :2OS6:$ULR.MAX.GROUP.2(*0011)
   ------------------------------ HISTORY      ------------------------------
   CRE-DATE   = 2014-10-12  ACC-DATE   = 2014-10-12  CHANG-DATE = 2014-10-12
   CRE-TIME   =   14:11:44  ACC-TIME   =   14:11:44  CHANG-TIME =   14:11:44
   ACC-COUNT  = 1           S-ALLO-NUM = 0
   ------------------------------ SECURITY     ------------------------------
   READ-PASS  = YES         WRITE-PASS = NONE        EXEC-PASS  = NONE
   USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
   AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2014-10-12
   DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
   SP-REL-LOCK= NO          ENCRYPTION = *NONE
   ------------------------------ BACKUP       ------------------------------
   BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
   MIGRATE    = ALLOWED
   #BACK-VERS = 0
   ------------------------------ ORGANIZATION ------------------------------
```

```
   FILE-STRUC = PAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
   IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
   AVAIL      = *STD
   WORK-FILE  = *NO         F-PREFORM  = *K          S0-MIGR    = *ALLOWED
   ------------------------------ ALLOCATION   ------------------------------
   SUPPORT    = PUB         S-ALLOC    = 34          HIGH-US-PA = 4
   EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
       2       6VS1.0       D3435
   NUM-OF-EXT = 2
0000000024 :2OS6:$ULR.MAX.GROUP.2(*0012)
   ------------------------------ HISTORY      ------------------------------
   CRE-DATE   = 2014-10-12  ACC-DATE   = 2014-10-12  CHANG-DATE = 2014-10-12
   CRE-TIME   =   14:16:00  ACC-TIME   =   14:16:00  CHANG-TIME =   14:16:00
   ACC-COUNT  = 1           S-ALLO-NUM = 0
   ------------------------------ SECURITY     ------------------------------
   READ-PASS  = YES         WRITE-PASS = NONE        EXEC-PASS  = NONE
   USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
   AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2014-10-12
   DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
   SP-REL-LOCK= NO          ENCRYPTION = *NONE
   ------------------------------ BACKUP       ------------------------------
   BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
   MIGRATE    = ALLOWED
   #BACK-VERS = 0
   ------------------------------ ORGANIZATION ------------------------------
   FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
   IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
   REC-FORM   = (V,N)       REC-SIZE   = 0
   AVAIL      = *STD
   WORK-FILE  = *NO         F-PREFORM  = *K          S0-MIGR    = *ALLOWED
   ------------------------------ ALLOCATION   ------------------------------
   SUPPORT    = PUB         S-ALLOC    = 9           HIGH-US-PA = 1
   EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
      2       6VS1.0       D3435
   NUM-OF-EXT = 2
 0000000024 :2OS6:$ULR.MAX.GROUP.2(*0013)
   ------------------------------ HISTORY      ------------------------------
   CRE-DATE   = 2014-10-12  ACC-DATE   = 2014-10-12  CHANG-DATE = 2014-10-12
   CRE-TIME   =   14:16:04  ACC-TIME   =   14:16:05  CHANG-TIME =   14:16:05
   ACC-COUNT  = 1           S-ALLO-NUM = 0
   ------------------------------ SECURITY     ------------------------------
   READ-PASS  = YES         WRITE-PASS = NONE        EXEC-PASS  = NONE
   USER-ACC   = ALL-USERS   ACCESS     = WRITE       ACL        = NO
   AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2014-10-12
   DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
   SP-REL-LOCK= NO          ENCRYPTION = *NONE
   ------------------------------ BACKUP       ------------------------------
   BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
   MIGRATE    = ALLOWED
   #BACK-VERS = 0
   ------------------------------ ORGANIZATION ------------------------------
   FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
   IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
   REC-FORM   = (V,N)       REC-SIZE   = 0
   AVAIL      = *STD
   WORK-FILE  = *NO         F-PREFORM  = *K          S0-MIGR    = *ALLOWED
------------------------------ ALLOCATION   ------------------------------
   SUPPORT    = PUB         S-ALLOC    = 9           HIGH-US-PA = 1
   EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
```

```
       2       6VS1.1       D3435
   NUM-OF-EXT = 2
:2OS6: PUBLIC:      4 FILES RES=        72 FRE=       66 REL=       60 PAGES
```

Wie im Gruppeneintrag festgelegt, sind alle Generationen dieser FGG mehrbenutzbar und durch ein Lesekennwort geschützt. Sie unterscheiden sich jedoch in den Eigenschaften, die nicht über den Gruppeneintrag festgelegt werden: ACC-COUNT, CRE-DATE, CHANG-DATE, EXPIR-DATE.

10.2.2 Dateigenerationen erstellen

Dateigenerationen werden wie „normale“ Dateien mit den Makros FILE oder CATAL bzw. mit dem Kommando CREATE-FILE-GENERATION katalogisiert. Auch die Dateiverarbeitung – hier: Erstellen der Datei – entspricht der „normalen“ Dateiverarbeitung. Der einzige Unterschied besteht in der Form des Dateinamens, der eine absolute oder relative Generationsnummer enthalten muss. Beim Katalogisieren oder Erstellen von Dateigenerationen muss der Benutzer beachten, dass die lückenlose Aufeinanderfolge der absoluten Generationsnummern erhalten bleiben muss – gleichgültig, ob er mit den absoluten oder mit relativen Generationsnummern arbeitet. Der Gruppeneintrag wird vom DVS automatisch aktualisiert.

Für einzelne Dateigenerationen einer FGG können die Attribute STOCLAS, IOPERF, IOUSAGE, DISKWR, S0MIGR, AVAIL, USRINFO, ADMINFO (mit Makros) bzw. STORAGE-CLASS, PERFORMANCE, USAGE, DISK-WRITE, S0-MIGRATE, AVAILABILITY, USER-INFORMATION, FILE-PREFORMAT, ADM-INFORMATION (mit Kommando) jeweils separat vergeben (und geändert) werden.

10.3 Dateigenerationen/Dateigenerationsgruppen löschen

Ebenso wie beim Erstellen von Generationen, dürfen auch beim Löschen keine „Löcher“ in der Folge der absoluten Generationsnummern entstehen. Ausgehend von der gewählten Generation dürfen daher nur alle jüngeren oder alle älteren Generationen gelöscht werden (Operand POS im ERASE-Makroaufruf, Operand DELETE im Kommando DELETE-FILE-GENERATION).

Wird durch eine Löschoperation die bisherige Bezugsgeneration für relative Generationsnummern gelöscht, wird die im Makro ERASE bzw. im Kommando DELETE-FILE-GENERATION angegebene Generation zur neuen Bezugsgeneration.
Die Felder des Gruppeneintrags werden entsprechend aktualisiert (BASE-NUM, FIRST-GEN, LAST-GEN).

Dateigenerationsgruppen lassen sich durch Angabe des Gruppennamens im Makro ERASE bzw. im Kommando DELETE-FILE-GROUP löschen. Gleichzeitig mit dem Gruppeneintrag werden auch alle noch katalogisierten Dateigenerationen gelöscht.

Im Makro ERASE (VERSION=1 / 2 / 3) können auch der Selektionsparameter TYPE=FGG oder Musterzeichen im Dateinamen genutzt werden, sodass sich ein ERASE-Makroaufruf auf mehrere Dateigenerationsgruppen auswirken kann.

Im Kommando DELETE-FILE-GROUP können auch Musterzeichen im Dateinamen genutzt werden, sodass sich der Kommandoaufruf auf mehrere Dateigenerationsgruppen auswirken kann.

Beispiel: Generationen löschen

Ausgangssituation: GROUP1 besteht aus den Generationen 6 bis 9, BASE-NUM=7

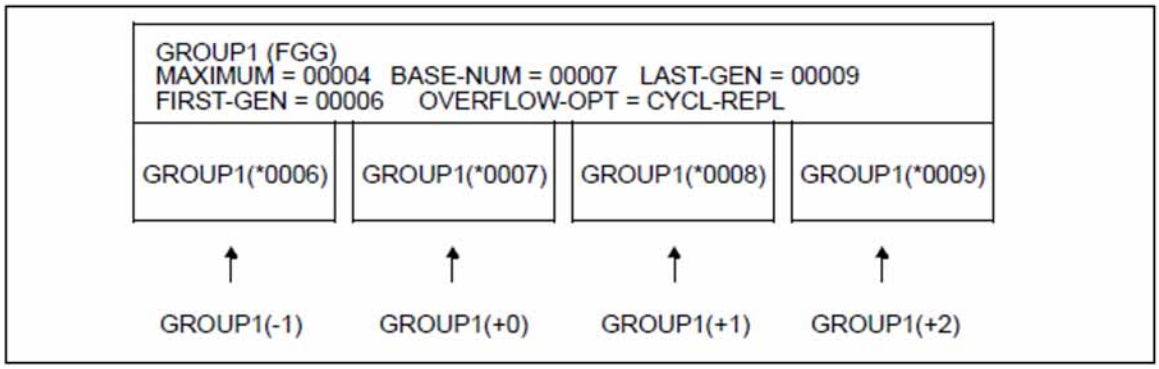


Bild 14: Dateigenerationen löschen (1)

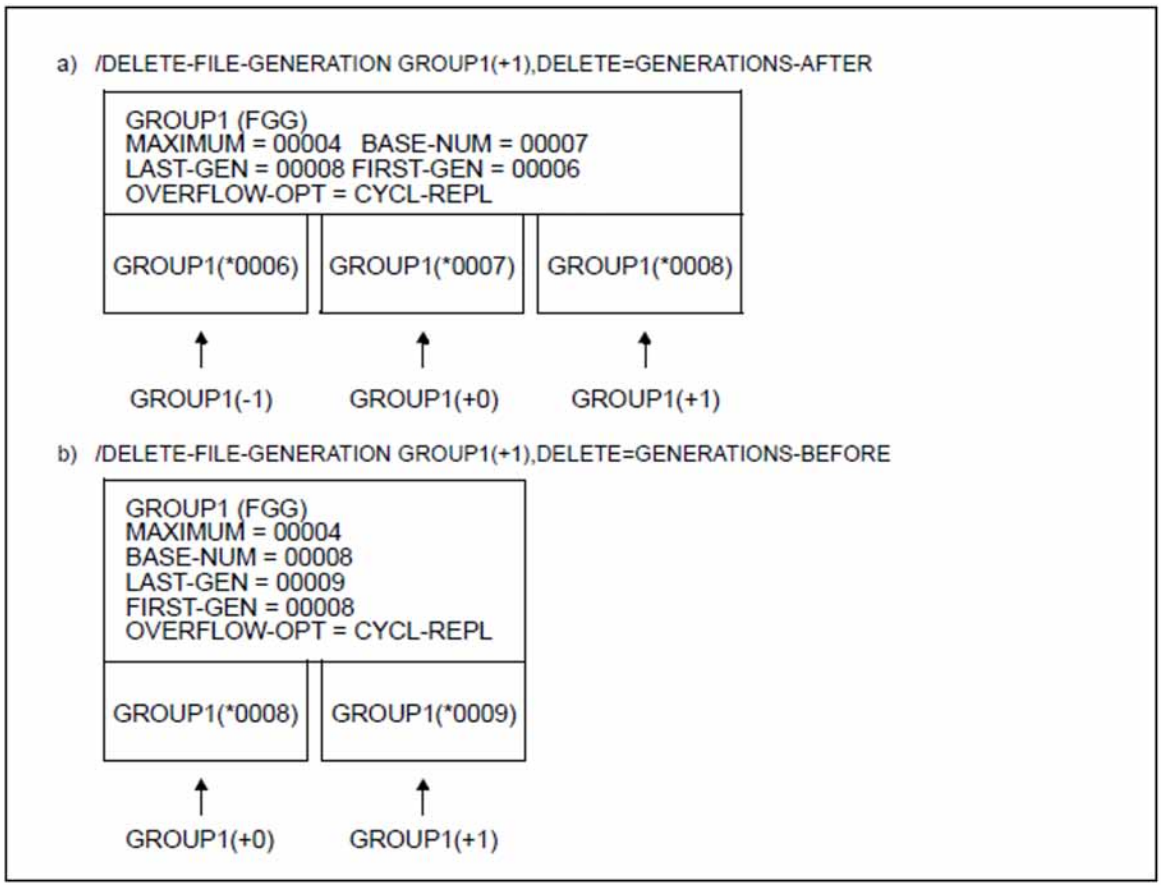


Bild 15: Dateigenerationen löschen (2)

10.4 Datenträger für Dateigenerationsgruppen

Dateigenerationsgruppen können auf gemeinschaftlichen wie auch auf privaten Datenträgern (Platten oder Magnetbändern) gespeichert werden. Bei der Wahl des Speichermediums sollten Sie berücksichtigen, welchem Zweck die FGG dient: wird diese zum Aufbau eines großen Datenpools genutzt, in dem z.B. Buchhaltungsdaten über mehrere Jahre abrufbereit gehalten werden, oder dient sie der Verwaltung von Sicherungskopien, die zwischen regelmäßig wiederkehrenden Verarbeitungen – wie z.B. der Aktualisierung von Stammdaten – oder bei der automatischen Dateisicherung in Rechenzentren angelegt werden?

> ¡ Wird eine FGG auf Privatplatten erstellt, müssen alle Generationen auf Privatplatte gespeichert werden. Im Gegensatz dazu ist eine gemischte Verwendung von Pubsets und Bändern möglich.

10.4.1 Pubsets

Wird eine FGG auf einem SF-Pubset eingerichtet, hat der Benutzer – wie bei anderen Dateien auch – keinen Einfluss darauf, wo die Generationen gespeichert werden, d.h. auf welchen Platten des Pubsets.

Beim Einrichten einer FGG auf einem SM-Pubset muss implizit über die Vergabe einer De-fault-Storage-Klasse oder explizit durch Angabe des Operanden WORKGRP (Makro) bzw. WORK-FILE-GROUP (Kommando) festgelegt werden, ob es sich um eine Gruppe von permanenten Generationen oder um eine Arbeits-Generationsgruppe handelt (Attribut WORK-FILE). Eine nachträgliche Änderung des Attributs ist nicht möglich.
Wenn den zugehörigen Generationen eine Storage-Klasse zugewiesen wird (z.B. bei Erstallokierung mit dem Makro FILE) oder die Storage-Klasse ausgetauscht wird, muss das Attribut WORK-FILE in der Storage-Klasse mit dem Attribut der Gruppe übereinstimmen.

10.4.2 Privatplatten

Die Verwendung privater Platten für Generationen einer FGG schließt die gleichzeitige Verwendung anderer Datenträger für Generationen dieser FGG aus.

Gruppeneintrag und Einträge für die Dateigenerationen werden in den F1-Kennsatz der Privatplatte übernommen. Sie können im Katalogeintrag mit dem Makro CATAL (Operand DISP=REUSE) bzw. mit den Kommandos CREATE-FILE-GROUP und MODIFY-FILE-GROUP-ATTRIBUTES (Operand OVERFLOW-OPTION=*REUSE-VOLUME) festlegen, dass zum Speichern einer neuen Generation der Datenträger herangezogen wird, auf dem die zu löschende älteste Generation steht. Erstreckt sich diese zu löschende Generation über mehrere Platten, wird die neue Generation nur auf der ersten Platte der Serie katalogisiert.

In Zusammenhang mit den o.g. Operanden muss Folgendes berücksichtigt werden:
der Speicherplatz der zu löschenden Generation wird erst freigegeben, *nachdem* die neue Generation eingerichtet wurde. Ist die Privatplatte vollständig belegt und versuchen Sie eine Generation mit dem Makro FILE bzw. dem Kommando CREATE-FILE-GENERATION einzurichten, kann für die Datei keine Primärzuweisung ausgeführt werden und der Makro- oder Kommandoaufruf wird wegen Speicherplatzmangels abgewiesen.

Im Gegensatz zu FGG auf Pubsets können FGG oder Generationen auf Privatplatten exportiert werden (mit dem Makro ERASE, Operanden CATALOG/DELETE-OR-EXPORT/VOLUME oder dem Kommando EXPORT-FILE) und später jederzeit wieder ins System importiert werden.

10.4.3 Magnetbänder / Magnetbandkassetten

Werden Dateigenerationen auf Bändern gespeichert, muss berücksichtigt werden, dass der Gruppeneintrag nur im Dateikatalog des Pubsets existiert, in dem die FGG und ihre Generationen katalogisiert wurden; er wird nicht auf Band übernommen. Die Dateigenerationen sind im HDR1-Kennsatz durch ihre Generationsnummer als Dateigenerationen gekennzeichnet.

Wie andere Banddateien können sich auch Dateigenerationen über mehrere Bänder erstrecken oder zu mehreren auf einem Band gespeichert werden. Wiederverwendung frei werdender Datenträger (DISP=REUSE bzw. OVERFLOW-OPTION=REUSE-VOLUME) ist allerdings nur möglich, wenn die Dateigenerationen nicht als File Set gespeichert werden, sondern nur als einzige Datei auf einem Band oder in einem Volume Set.

Wie FGG/Generationen auf Privatplatten können auch Generationen auf Band exportiert und später wieder importiert werden.

10.5 Dateigenerationen / Dateigenerationsgruppen exportieren

Es können nur Dateigenerationen bzw. FGG auf privaten Datenträgern exportiert werden. Exportieren heißt: der entsprechende Eintrag im Dateikatalog wird gelöscht, der Speicherplatz wird jedoch nicht freigegeben, die Daten bleiben erhalten.

Das Exportieren erfolgt über das Kommando EXPORT-FILE oder den Makro ERASE mit den Operanden CATALOG bzw. DELETE-OR-EXPORT und VOLUME. Der VOLUME-Operand kann nur für FGG auf Privatplatten angegeben werden: es werden dann die Katalogeinträge der Generationen gelöscht, die auf der entsprechenden Privatplatte gespeichert sind.

Beim Exportieren von Dateigenerationen ist – wie beim Löschen – darauf zu achten, dass keine Lücken in der Folge der noch katalogisierten Generationen entstehen.

Beispiel: FGG auf Privatplatten exportieren

Auf zwei privaten Platten ist eine Dateigenerationsgruppe wie folgt verteilt:

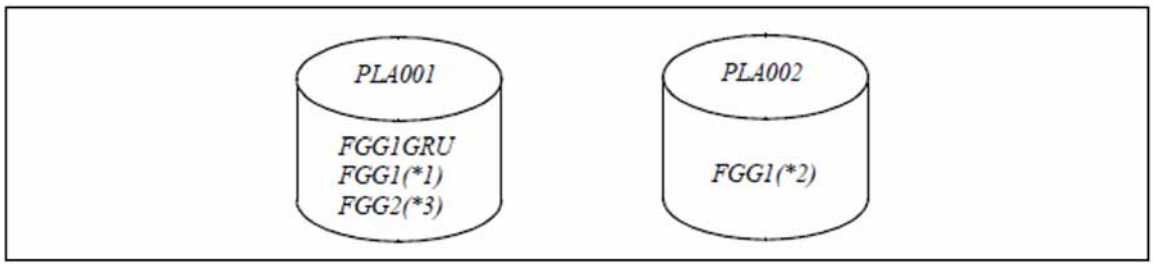


Bild 16: Dateigenerationsgruppe exportieren

GRU=Gruppeneintrag

Das Kommando EXPORT-FILE VOLUME=PLA002 löscht den Eintrag der Generation DGG1(*2) aus dem Dateikatalog, die Generationen *1 und *3 bleiben nach wie vor katalogisiert: die Dateigenerationsgruppe hat ein „Loch“ bzgl. der Katalogeinträge; auf den Platten ist die Dateigenerationsgruppe vollständig.

10.6 Dateigenerationen / Dateigenerationsgruppen importieren

Dateigenerationsgruppen und Dateigenerationen auf Privatplatten, Magnetbändern oder Magnetbandkassetten können jederzeit in ein System importiert werden.

FGG oder Generationen auf Privatplatten können entweder einzeln über die Makros CATAL bzw. FILE (jeweils mit STATE=FOREIGN), gruppenweise mit dem Makro IMPORT oder einzeln und gruppenweise mit dem Kommando IMPORT-FILE importiert werden.

> ¡ Der Makro CATAL kann nur FGG importieren, nicht jedoch Generationen; der Makro FILE kann nur Generationen importieren, nicht jedoch FGG.

Der Makro IMPORT und das Kommando IMPORT-FILE erstellen die Katalogeinträge aus den Einträgen im F1-Kennsatz der Privatplatte. Dateigenerationen auf Bändern können nur über FILE-Makroaufruf, nicht mit dem Makro IMPORT importiert werden. Bei Verwendung des Kommandos IMPORT-FILE können Dateigenerationen auf Bändern und Privatplatten verarbeitet werden.

Beim Importieren einer FGG oder von Teilen einer FGG ist immer darauf zu achten, dass zunächst der Gruppeneintrag erstellt werden muss, bevor die erste Dateigeneration katalogisiert werden kann. Für die Anwendung des Makros IMPORT bzw. des Kommandos IMPORT-FILE bei Privatplatten-FGGs heißt dies: zuerst muss die Privatplatte angefordert werden, die den Gruppeneintrag enthält, oder es muss ein Gruppeneintrag mit dem Makro CATAL bzw. mit dem Kommando CREATE-FILE-GROUP erstellt werden.

Für das Importieren mit den Makros CATAL und FILE gilt Folgendes: Zunächst muss der Gruppeneintrag mit CATAL-Makro und der Angabe STATE=FOREIGN, DEVICE und VOLUME erstellt werden. Danach müssen die Generationen einzeln mit FILE katalogisiert werden – unter Angabe der Operanden STATE=FOREIGN, DEVICE und VOLUME.

Auch beim Import von FGG/Dateigenerationen gilt: es dürfen keine Lücken in der Generationenfolge entstehen.

Beim Importieren von privaten Platten, auf denen Dateigenerationsgruppen stehen, ist zu beachten, dass ein Aufruf des Makros IMPORT bzw. des Kommandos IMPORT-FILE nur solche Generationen katalogisiert, deren Gruppeneintrag entweder auf der bezeichneten Platte oder bereits im Systemkatalog steht. Für eine Dateigenerationsgruppe, die auf mehrere Platten verteilt und noch nicht katalogisiert ist, hat dies folgende Auswirkungen: Bringt man zunächst eine Platte ein, die den Gruppeneintrag nicht enthält, und erst dann die Platte mit dem Gruppeneintrag, fehlen anschließend die Katalogeinträge der Generationen, die auf der ersten Platte stehen. Abhilfe: je ein erneuter Makro IMPORT bzw. ein erneutes Kommando IMPORT-FILE für die betreffenden Datenträger oder je ein FILE-Makroaufruf (Operand STATE=FOREIGN) bzw. ein Kommando IMPORT-FILE für die nicht katalogisierten Generationen.

Beim Import von Bandgenerationen werden zusätzlich folgende Attribute vom FGG-Index in den Katalogeintrag der Generationen übernommen:
AUDIT, BACKUP, DESTROY, LARGE, MIGRATE, NUM-OF-BACKUP-VERS und SHARE sowie der Indikator für verschlüsselte Kennwörter.

Beispiel: FGG auf Privatplatten importieren

Auf drei privaten Platten sind zwei Dateigenerationsgruppen wie folgt verteilt:

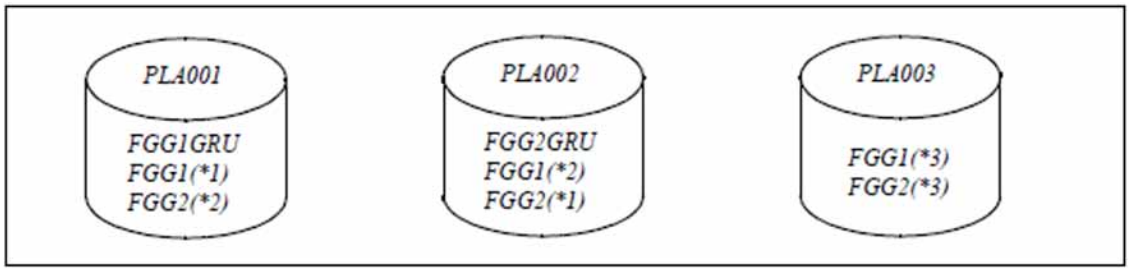


Bild 17: Dateigenerationsgruppe importieren

GRU=Gruppeneintrag

Dateigenerationen und Gruppeneinträge sind nicht im System katalogisiert.

Nach Ausführung der Kommandos

```
/IMPORT-FILE VOLUME=PLA001,DEVICE-TYPE=D3435
/IMPORT-FILE VOLUME=PLA002,DEVICE-TYPE=D3435
/IMPORT-FILE VOLUME=PLA003,DEVICE-TYPE=D3435
```

sind folgende Generationen bzw. Gruppeneinträge katalogisiert:

```
FGG1 (FGG)  FGG2 (FGG)
FGG1 (*1)   FGG2(*1)
FGG1 (*2)   FGG2(*3)
FGG1 (*3)
```

Die Generation FGG2(*2) lässt sich nachträglich z.B. mit dem Kommando IMPORT-FILE einbringen.

Man beachte, dass es für die Kommandos keine Reihenfolge gibt, bei der alle Generationen katalogisiert würden!

10.7 Dateigenerationsgruppen reservieren

Es ist nicht immer auszuschließen, dass verschiedene Generationen einer Dateigenerationsgruppe gleichzeitig von verschiedenen, parallel laufenden Aufträgen verarbeitet, erstellt oder gelöscht werden. Solche unkontrollierten gleichzeitigen Zugriffe können sehr leicht zu Inkonsistenz innerhalb der Generationenfolge führen, aber auch zu Daten-Inkonsistenz, wenn die Dateigenerationen aus aufeinander aufbauenden Verarbeitungsschritten resultieren.

Ein Mittel, die durch parallele Zugriffe entstehenden Probleme zu vermeiden, ist das Reservieren der Dateigenerationsgruppe mit dem Kommando SECURE-RESOURCE-ALLOCATION (Operand FILE). Dabei werden nicht die Datenträger für weitere Zugriffe gesperrt; es wird lediglich Zugriff auf den Gruppeneintrag und die Dateigenerationen unterbunden. Von dieser Reservierungsmöglichkeit sollte immer dann Gebrauch gemacht werden, wenn mehrfacher gleichzeitiger Zugriff auf eine FGG nicht ausgeschlossen werden und dieser Zugriff zu Inkonsistenzen führen kann. Eine Reservierung von FGG/Dateigenerationen ist z.B. nicht nötig, wenn die Generationen nur gelesen werden können.

Das folgende Beispiel zeigt, welche Probleme ohne Reservierung bei gleichzeitigem Zugriff verschiedener Aufträge auftreten können.

Beispiel: Parallelzugriff auf eine FGG

Ausgangssituation: die FGG GROUP1 besteht aus maximal 3 Generationen, zurzeit den Generationen 4, 5 und 6; die Generation 5 ist als Bezugspunkt für relative Indizierung definiert (BASE-NUM=5).
Parallel zueinander werden die Aufträge 0001 und 0002 gestartet, die folgende Aktionen auslösen:

Auftrag 0001:

```
/CREATE-FILE-GEN GEN-NAME=GROUP1(*7)
/MODIFY-FILE-GROUP-ATTRIBUTES GROUP-NAME=GROUP1,
     GENERATION-PARAMETER=*GENERATION-PARAMETER(BASE-
     NUMBER=ABSOLUTE(NUMBER=0))
```

Auftrag 0002:

```
/ADD-FILE-LINK LINK-NAME=EINGABE,FILE-NAME=GROUP1(-1)
```

Wenn das Kommando ADD-FILE-LINK des zweiten Auftrags eingegeben wird, nachdem das Kommando CREATE-FILE-GENERATION wirksam geworden ist, erhält der Benutzer die Fehlermeldung „Datei nicht vorhanden“; hätte er im Kommando ADD-FILE-LINK die absolute Generationsnummer (*4) angegeben, erhielte er ebenfalls diese Fehlermeldung.

Ist vor dem Kommando ADD-FILE-LINK auch das Kommando MODIFY-FILE-GROUP-ATTRIBUTES wirksam geworden, findet das DVS zwar eine Dateigeneration GROUP1(-1), es ist jedoch nicht die von Auftrag 0002 erwartete Generation (*4), sondern die Generation (*6); die anschließende Dateiverarbeitung führt zu falschen Ergebnissen.

11 OPEN-Verarbeitung

Jedes Programm, das mit einer Datei arbeiten soll, muss bei seinem Ablauf in einem Auftrag folgende Aktionen auslösen:

- Wenn die Datei noch nicht katalogisiert und mit Speicherplatz versehen ist: Datei einrichten (mit dem Makro FILE oder dem Kommando CREATE-FILE)
- Datei eröffnen (mit einem OPEN-Makroaufruf): das System prüft, ob der aufrufende Auftrag auf die Datei zugreifen darf und bereitet den eigentlichen Zugriff zum Dateiinhalt vor
- Zugreifen auf den Dateiinhalt (mit den Zugriffsmethoden des DVS): das System überträgt auf Anforderung Daten zwischen dem aufrufenden Auftrag und der Datei
- Datei schließen (mit einem CLOSE-Makroaufruf): das System trennt den aufrufenden Auftrag logisch von der Datei, danach weist es Anforderungen zur Datenübertragung zurück

Ein Auftrag kann eine Datei beliebig oft eröffnen, auf ihren Inhalt zugreifen und sie schließen. Da Öffnen und Schließen einer Datei zeitaufwändige Aktionen sind, sollte ein Auftrag Dateien erst dann eröffnen, wenn er sie bearbeiten kann, und schließen, wenn die Bearbeitung beendet ist.

11.1 Informationen für die OPEN-Verarbeitung

Welche Informationen benötigt das DVS beim Eröffnen einer Datei?

- Welche Datei ist zu eröffnen?
- Erfüllt der aufrufende Auftrag die Bedingungen, die durch die Schutzmerkmale der Datei gegeben sind (z.B. Kennwörter)?
  Siehe dazu im Kapitel „Datei- und Datenschutz“.
- Mit welcher Zugriffsmethode ist die Datei zu bearbeiten? Wie ist die Datei organisiert (Blockgröße, Satzlänge usw.)?
- Ist die Datei als Eingabedatei oder als Ausgabedatei zu eröffnen? Wenn die Datei als Ausgabedatei zu eröffnen ist: soll sie ersetzt oder erweitert werden?
- In welchem Teil des Adressraums sind Datenpuffer einzurichten, im Benutzerspeicher (Klasse-6-Speicher) oder im Systemspeicher (Klasse-5-Speicher)?
- Wie sind die Datensätze an den Auftrag zu übergeben und von ihm zu übernehmen? Ist jeweils die Adresse zu übergeben, an der der aktuelle Satz im Puffer beginnt (Ortungsbetrieb / Locate Mode), oder ist der Satz in einem Benutzerbereich zu übergeben (Übertragungsbetrieb / Move Mode)?
- Dürfen zu dieser Datei gleichzeitig andere Aufträge zugreifen (Shared-Update-Verarbeitung)?
- Wenn die Datei auf privaten Datenträgern steht oder erstellt werden soll: welche Datenträger werden benötigt?
- An welcher Adresse ist der Programmlauf fortzusetzen, wenn ein Fehler oder ein anderes Ereignis auftritt, auf das das Programm reagieren muss?

Woher bezieht das DVS die erforderlichen Informationen?

- aus dem **TFT-Eintrag**, der den gleichen Dateikettungsnamen enthält wie der Dateisteuerblock des Programms (FCB; der TFT-Eintrag wird erzeugt mit dem Kommando ADD-FILE-LINK, Operand LINK-NAME bzw. FILE-Makro, Operand LINK). Die TFT ist die zentrale Verwaltungstabelle des DVS
- aus dem **Dateisteuerblock** (FCB), d.h. der Dateierklärung des Programms (Falls in der TFT keine entsprechende Angabe steht bzw. kein TFT-Eintrag mit dem gleichen Dateikettungsnamen existiert). Falls kein TFT-Eintrag vorhanden ist, muss im Dateisteuerblock der Dateiname eingetragen sein (Makro FCB, Operand FILE)
- aus dem **Katalogeintrag** der Datei
- aus der Kennworttabelle des Auftrags. Ein Kennwort kann im Makro FCB, Operand PASS bzw. im Kommando ADD-PASSWORD angegeben werden
- aus dem OPEN-Makroaufruf den OPEN-Modus (OPEN-Makroaufruf hat Vorrang vor den Angaben in TFT und FCB)

Für Angaben, die in mehr als einer Quelle enthalten sind, gilt Folgendes:

- Angaben zum OPEN-Modus im OPEN-Aufruf haben Vorrang vor TFT und FCB (es gibt keinen Katalogeintrag für OPEN-Modus)
- Angaben im TFT-Eintrag haben Vorrang vor Angaben im Dateisteuerblock und im Katalogeintrag
- Angaben im Dateisteuerblock haben Vorrang vor Angaben im Katalogeintrag

- Wurde im FILE- oder FCB-Makro der Nulloperand verwendet bzw. im Kommando ADD-FILE-LINK bei einem Operanden der Operandenwert *BY-CATALOG eingestellt, werden die entsprechenden Felder aus dem Katalogeintrag übernommen.

  Felder, die aus dem TFT-Eintrag in den FCB übernommen werden:

- BLIM
- BLKSIZE*)
- BLKCTRL*)
- BUFOFF*)
- CHAINIO
- CHKPT
- CLOSE
- CLOSMSG
- CODE
- DUPEKY
- FCBTYPE*)
- FILEFSEQ*)
- IOPERF*)
- IOUSAGE*)
- KEYLEN*)
- KEYPOS*)
- LABEL
- LARGE_FILE
- LINK
- LOCKENV
- LOGLEN*)
- POOLLNK
- OPEN
- OVERLAP
- PAD
- RECFORM*)
- RECSIZE*)
- RETPD
- SECLEV
- SHARUPD
- TAPEWR
- TPMARK
- TRANS
- VALLEN*)
- VALPROP*)
- WRCHK
- WROUT

Die mit *) gekennzeichneten Felder können als Nulloperanden bzw. mit *BY-CATALOG angegeben werden.

- Informationen über die Schutzfrist einer bestehenden Datei entnimmt das DVS immer direkt dem Katalogeintrag oder dem HDR1-Kennsatz
- Wenn Angaben aus der TFT (Makro FILE, Kommando ADD-FILE-LINK) bzw. dem FCB unverträglich sind mit dem Inhalt des Katalogeintrags, bricht das DVS das Eröffnen mit einem Fehlercode ab

Dateisteuerblock (FCB – File Control Block)

Der Dateisteuerblock beschreibt die Datei bzw. die Dateimerkmale im Benutzerprogramm. Er lässt sich in folgenden Schritten erstellen bzw. verändern:

- bei der Programmerstellung, d.h. vor dem Eröffnen, legt der FCB-Makroaufruf den Dateisteuerblock an und versieht ihn wahlweise mit Informationen
- im Programmlauf vor dem Eröffnen der Datei durch Versorgen von Feldern im Dateisteuerblock (FCB) mit aktuellen Werten. Ein Hilfsmittel dazu sind die Makroaufrufe IDFCB und IDFCBE, die eine Dsect für den FCB generieren
- während des Eröffnens kann der Benutzer den Dateisteuerblock nach bestimmten Verarbeitungsschritten der OPEN-Routine überprüfen und gegebenenfalls ändern (Makro EXLST, Operanden OPENX und OPENZ)

Solange eine Datei eröffnet ist, enthält der Dateisteuerblock alle Informationen, die den aktuellen Zustand der Datei beschreiben. Für den Benutzer sind diese Informationen von Bedeutung, wenn ein Fehler auftritt und im Programm analysiert werden soll (siehe Makro FCB, Operand EXIT, und Makro EXLST). Der Dateisteuerblock enthält dann einen DVS-Fehlerschlüssel und eine Kennzeichnung des Fehlerausgangs.

Informationsquellen des DVS

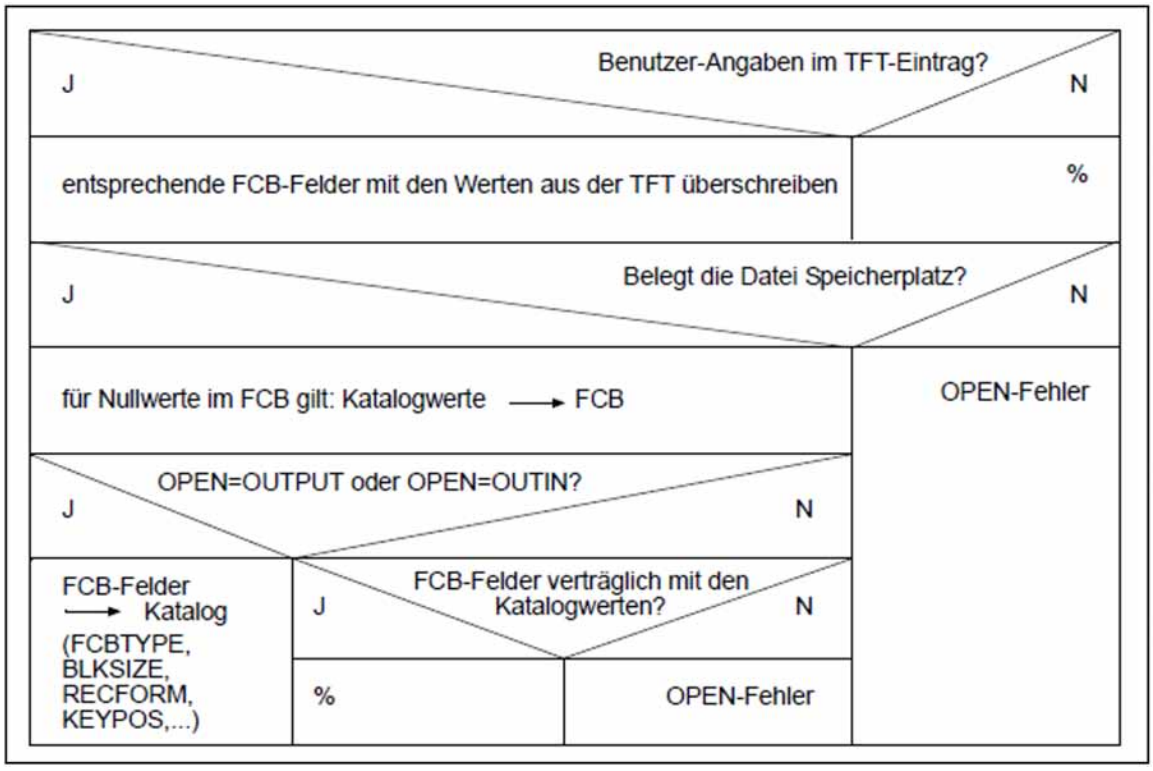


Bild 18: Informationsquellen des DVS

11.2 Ablauf der OPEN-Verarbeitung

Der Ablauf der OPEN-Verarbeitung variiert in geringem Ausmaß, abhängig davon, ob der Benutzer die XS-Schnittstelle (31-Bit-Adressierung) nutzt. Im Folgenden wird der Ablauf für die XS-OPEN-Verarbeitung beschrieben.

Die Besonderheiten der 24-Bit-Schnittstelle sind im Abschnitt „XS-/Nicht-XS-Schnittstelle“ beschrieben.

Das DVS führt folgende Aktionen durch:

- Dateisteuerblock vervollständigen aus den Quellen TFT und Katalogeintrag
- falls nötig private Datenträger anfordern
- den Dateisteuerblock mit den Routinen der Zugriffsmethoden verbinden. Den FCB mit den Adressen der logischen Routinen (TU-Logicals) versorgen, die das Blocken und Entblocken der Datensätze übernehmen
- privilegierten internen Dateisteuerblock (TPR-FCB) aufbauen: der Benutzer hat zu diesem Steuerblock keinen Zugriff; er wird vom DVS intern verwendet
- Pufferbereiche zuweisen
- Katalogeintrag aktualisieren
  Insbesondere bei Node-Files können die Dateiattribute CHANGE-DATE, FILESIZE, Highest-Used-Page (LPP) und LBP veraltet sein, wenn ein fremdes System die Dateien geändert hat. Diese Werte werden im Rahmen des OPEN automatisch aktualisiert. Hierbei ist zu beachten, dass bei mehreren aufeinander folgenden OPEN (ohne CLOSE) die Aktualisierung des CEs nur beim ersten OPEN stattfindet. Die folgenden OPEN erhalten die aktuellen Werte des LPP und LBP im FCB zugestellt.
  Eine Aktualisierung des Katalogeintrages kann auch manuell mit IMPORT-NODE-FILE REPLACE = *NODE-FILE-UPDATE durchgeführt werden. Hierbei sind die aktualisierten Werte bei SAM-Node-Files für FILESIZE und LPP nur geschätzt.

> **Hinweis**
>
> Der Katalogeintrag einer Datei ist erst vollständig, wenn die Datei als Ausgabedatei eröffnet und wieder geschlossen wurde. Daher unterscheidet man zwischen katalogisierten und existenten Dateien.
> Beziehen sich DVS-Makros z.B. auf katalogisierte Dateien, bedeutet dies, dass ein Katalogeintrag für die Datei bestehen muss.
> Beziehen sich DVS-Makros auf existente Dateien, müssen diese mindestens einmal als Ausgabedateien eröffnet worden sein. Sie können jedoch leer sein, d.h. keine für den Benutzer zugreifbaren Datensätze enthalten.
> Erst zum CLOSE-Zeitpunkt wird der zuletzt aktuelle Seitenzeiger („Last Page Pointer“) und der „Last Byte Pointer“ übertragen.
> Der Last Byte Pointer ist nur für PAM-Dateien sowie SAM-Node-Files, die im UPAM-RAW-Modus geöffnet werden, relevant.

Der OPEN-Ablauf für eine Platten- oder Banddatei umfasst im Wesentlichen die folgenden Aktionen:

- Suche nach einem TFT-Eintrag mit dem Dateikettungsnamen, der im Dateisteuerblock enthalten ist. Der Dateikettungsname stammt entweder aus einem FCB-Makroaufruf oder wurde direkt in den Steuerblock geschrieben. Jeder FILE- oder ADD-FILE-LINK-Aufruf erzeugt bzw. CHNGE- oder CHANGE-FILE-LINK aktualisiert in der TFT einen Eintrag. Ein solcher Eintrag enthält alle Parameter zur Ergänzung der Dateisteuerblöcke und stellt über den Kettungsnamen (LINK-Parameter) die Verknüpfung zwischen Benutzerprogramm und Datei her. Die Suche nach einem TFT-Eintrag entfällt, wenn im FCB der Operand LINK einen Dateikettungsnamen aus 8 Leerzeichen (8X'40'; 8C'_') enthält.

  Aus dem korrespondierenden Eintrag in der TFT werden die entsprechenden Felder im Dateisteuerblock aktualisiert. Falls kein TFT-Eintrag mit dem angegebenen Dateikettungsnamen existiert bzw. falls der Dateikettungsname im FCB aus 8 Leerzeichen (8X'40'; 8C'_') besteht, wird während der OPEN-Verarbeitung ein TFT-Eintrag angelegt. Zum CLOSE-Zeitpunkt wird dieser TFT-Eintrag vom System automatisch wieder freigegeben (impliziter Release). Nur mit FILE (Operand LINK) erzeugte TFT-Einträge bleiben bis zu einem expliziten Release (Kommando REMOVE-FILE-LINK oder Makro RELTFT) erhalten.

  Felder, die im FILE-Makroaufruf als Nulloperanden angegeben wurden, haben in der TFT den Eintrag „BY-CATALOG“. Die entsprechenden Felder im FCB werden später mit Informationen aus dem Katalogeintrag vervollständigt.
- Komplettieren des Dateinamens. Der Dateiname wird, wenn noch nicht in der kompletten Form vom Benutzer angegeben, um die Katalogidentifikation „Catid“ und die Benutzerkennung „Userid“ ergänzt und in der Form „catid:$userid.dateiname“ im zugehörigen TFT-Eintrag und im FCB hinterlegt.
  Wird keine Userid angegeben, so wird die Benutzerkennung des Kommandos SET-LOGON-PARAMETERS verwendet. Wird keine Catid angegeben, so wird die Standard-Katalogkennung der Benutzerkennung verwendet.
- Einlesen des Katalogeintrages der angeforderten Datei. Ist die Datei weder unter der explizit angegebenen Catid und Userid noch unter der Standardkennung eines Standardkatalogs zu finden, wird der OPEN-Vorgang mit Fehleranzeige beendet. Ist im FCB-Aufruf der Operand OPTION=GLODEF angegeben, kommt es zu keinem Fehler, sondern es wird ein Secondary Read durchgeführt.

  Ein Beispiel für den „Secondary Read“, der über den FCB-Operanden OPTION=GLODEF ermöglicht wird, ist auf "Zugriff über die System-Standardkennung" zu finden.

  Soweit möglich werden die nicht definierten Felder des FCB mit Informationen aus dem dazugehörigen Katalogeintrag versorgt. Für Dateien mit dem Status FOREIGN (Banddateien im Dateiaustausch) werden die Parameter aus den Bandkennsätzen übernommen.
- Vergleich der Datenträgerliste des Katalogeintrags mit der in der TFT vermerkten Datenträgerliste. Sind keine Vermerke in der TFT enthalten, werden sie mithilfe der Katalogangaben aufgebaut und die benötigten Datenträger über die Geräteverwaltung angefordert.

- Bandgeräte zuweisen.

  Ist für die Datei ein TSET definiert, vergleicht das DVS die Datenträgerlisten von TFT-Eintrag und zugehörigem TST-Eintrag (Tape-Set-Table). Dabei geht es davon aus, dass die Datei auf dem Band beginnt, das in der TST als aktuelles Band geführt wird. Enthält die TFT-Datenträgerliste die aktuelle VSN (Archivnummer), werden die ihr vorangehenden Archivnummern der TFT-Datenträgerliste nicht in den Katalogeintrag übernommen. Enthält die TFT-Datenträgerliste die aktuelle VSN nicht, ersetzen die aktuelle und alle ihr in der TST-Liste folgenden Archivnummern die TFT-Datenträgerliste und werden in den Katalogeintrag übernommen.

  Ist für die Datei kein TSET definiert, im TFT-Eintrag jedoch eine temporäre Datenträgerliste enthalten (TVSN), weist das DVS entsprechend dieser TVSN das benötigte Band zu. Die Datenträgerliste im Katalogeintrag wird ignoriert. Der OPEN wird abgewiesen, wenn die Datei eine Ausgabedatei ist. Ist kein TSET definiert, aber im FILE-Makroaufruf VSEQ angegeben worden, darf die Datei nicht mit OPEN INPUT/EXTEND eröffnet werden. Es wird das erste oder das letzte Band aus der TFT-Datenträgerliste zugewiesen.

  Sind weder TSET noch TVSN oder VSEQ definiert, wird die Datenträgerliste des Katalogeintrags mit der des TFT-Eintrags in Einklang gebracht und das erste bzw. letzte Band der Liste zugewiesen.

- Aufbau des internen privilegierten Dateisteuerblocks (TPR-FCB) und Initialisierung mit den Feldinhalten des FCBs und des Katalogeintrags.

  Der ursprüngliche FCB (vor dem OPEN) wird mit den Feldinhalten in einem Sicherstellungsbereich am Ende des TPR-FCB abgespeichert.

- Sprung zum OPENX-Ausgang des EXLST-Makroaufrufs (falls im Programm vorgesehen). Das Programm kann an dieser Stelle die bis dahin eingestellten Werte überprüfen. Ändert das Programm die Felder des Dateisteuerblocks, so wird die Verarbeitung mit diesen Werten fortgesetzt. Bei neu anzulegenden Dateien werden diese Werte außerdem in den Katalogeintrag und in die Kennsätze übernommen. Der Dateiname, der Linkname und der Shared-Update-Modus können nicht mehr geändert werden. Die Verarbeitung wird mit dem EXRTN-Makroaufruf fortgesetzt.

- Sprung zum OPENZ-Ausgang des EXLST-Makroaufrufs (falls im Programm vorgesehen). An dieser Stelle kann das Programm den Dateisteuerblock noch einmal verändern, diese Änderungen werden jedoch nicht in den Katalogeintrag übernommen. So ist es zum Beispiel möglich, eine Datei mit einer anderen als der im Katalog eingetragenen Zugriffsmethode zu bearbeiten. Der Dateiname, der Linkname, der OPEN-Modus und der Shared-Update-Modus können nicht mehr geändert werden. Die Verarbeitung wird mit dem EXRTN-Makroaufruf fortgesetzt.

- Anhand der IO-Performance-Angaben und der eingerichteten Cache-Konfiguration festlegen, ob die Datei über einen Cache bearbeitet werden soll oder nicht. Dabei wird folgendermaßen verfahren:

  Ist die Datei bereits von einem anderen Auftrag geöffnet, werden die aktuell eingestellten Attribute auch für diesen OPEN übernommen.

  Handelt es sich um den ersten OPEN auf diese Datei, werden die statischen Performance-Attribute (PERFORMANCE, USAGE und DISK-WRITE) gegen die Cache-Eigenschaften (z.B. bei DISK-WRITE=*IMMEDIATE und USAGE=*WRITE/ *READ-WRITE wird Schreib-Caching nur bei einem schreibsicheren Cache zugelassen) geprüft und die danach eingestellten Werte im Katalogeintrag hinterlegt. Dabei kann der Benutzer über Angaben in der TFT oder im FCB die im Katalogeintrag definierten statischen PERFORMANCE- und USAGE-Werte in Richtung geringerer Wertigkeit dynamisch verändern und somit das Caching für diese eine Dateiverarbeitung ausschalten; die im Katalogeintrag hinterlegten Attribute bleiben dabei unverändert.

- Überprüfen der Benutzerangaben anhand des Katalogeintrags der Datei. (Datei muss Eingabedatei sein, d.h. OPEN-Mode nicht OUTPUT, nicht OUTIN)

- FCB-Werte in den Katalogeintrag übernehmen. Ist die Datei, auf die sich der FCB bezieht, eine Ausgabedatei (OPEN OUTPUT/OUTIN), werden die FCB-Werte in den Katalogeintrag übernommen.

- Überprüfen der Zugriffsangaben bzgl. Verfallsdatum, Zugriffsrechte und Kennwörter für Dateien. Diese Überprüfung wird durchgeführt, wenn die Datei bereits erstellt war.
- Bei der Neuerstellung der Datei (OPEN-Modus OUTPUT oder OUTIN) auf einem SM-Pubset überprüfen, ob die Eigenschaften des derzeitigen Ablageortes verträglich sind mit den geforderten Datei-Eigenschaften (z.B. Dateiformat BLKCTRL= PAMKEY). Falls nicht, muss eine entsprechende Umallokierung der Datei (ohne Datentransfer) auf einen geeigneten Ablageort angestoßen werden.
- Aufbauen bzw. Überprüfen der Kennsätze bei Magnetbandverarbeitung (siehe auch "Kennsatz-Verarbeitung beim Eröffnen von Banddateien"). Bei Neuerstellung einer Banddatei werden die Dateianfangskennsätze HDR1, HDR2 und HDR3 mit Werten versehen und vor den ersten Block der Datei geschrieben. Bei lesenden Zugriff wird der Inhalt dieser Kennsätze zur Identifikation der Datei und zur Überprüfung der Zugriffsberechtigung benutzt. Voraussetzung zur ordnungsgemäßen Verarbeitung der Kennsatzinformationen ist ihr normgerechter Aufbau nach DIN 66029. Enthält die Banddatei keine Standardkennsätze nach o.g. Normvorschrift, haben die Benutzerprogramme selbst die Überprüfung der Banddatei zu übernehmen.
- Zuweisen der Ein-/Ausgabebereiche bzw. Validierung der Pufferadressen, falls diese vom Benutzer angegeben wurden.
- Laden der Verbindungsroutinen (TU-Logicals) bei Verwendung der Zugriffsmethoden SAM oder ISAM.
- Verbindung zwischen Dateisteuerblock und Zugriffsmethoden herstellen, die Adressen der logischen Routinen werden in den FCB übernommen.
- Aktualisieren und Rückschreiben des Katalogeintrags, falls erforderlich.

Hinweise

Tritt nach der Umallokierung ein Fehler während des OPEN auf, ist die Datei anschließend zerstört.

Außerdem kann sich bei einer Umallokierung die Extent-Liste während des OPEN ändern, d.h. der Makro FSTAT bzw. das Kommando SHOW-FILE-ATTRIBUTES liefern nach dem OPEN eine andere Information als vorher.

Kann kein geeigneter Ablageort innerhalb des SM-Pubsets gefunden werden, wird der OPEN mit dem Returncode DMS0D80 abgewiesen.

11.3 OPEN in Konfigurationen mit Dateien > 32 GB

Die Schnittstelle OPEN prüft, ob Dateierweiterungen über 32 GB hinaus und das Erstellen oder Zugriffe auf Dateien >= 32 GB zulässig sind.

Hierbei gibt es zwei Aspekte:

1. Abweisen des Zugriffs auf oder der Erzeugung von großen Dateien für Zugriffsmethoden, die eine Bearbeitung von großen Dateien nicht gestatten.
2. Kennzeichnen, dass ein Programm Dateien >= 32 GB erzeugen bzw. öffnen kann.

Unverträgliche Schnittstellenvarianten

Schnittstellen, an denen 3-Byte-Blocknummern verwendet werden, sind prinzipiell nicht in der Lage, mit Dateien >= 32 GB zu arbeiten. Es handelt sich hier um folgende Fälle:

- Sämtliche Dateien im Key-Format (BLKCTRL=PAMKEY):
  Die logischen Blocknummern im Pamkey sind nur 3 Byte breit.
- 24-Bit-Schnittstelle von UPAM:
  Das Feld für die logischen Blocknummern in den UPAM-Parameterlisten und im TU-FCB ist nur 3 Byte breit.
- 24-Bit-Schnittstelle von SAM:
  Hier sind die logischen Blocknummern als Teil der Wiedergewinnungsaddresse betroffen.
- 24-Bit-Schnittstelle von ISAM

In allen oben aufgeführten Fällen gilt:

- Der Zugriff auf Dateien >= 32 GB wird mit dem Returncode `X'0000D9D'` oder `X'00000D00'` abgewiesen, abhängig von der Größe des für die Datei allokierten Speicherplatzes.
- Die Überschreitung einer Dateigröße von 32 GB durch Sekundärallokierung wird unterbunden.

Semantische Inkompatibilitäten

Es kann nicht ausgeschlossen werden, dass Anwendungen zwar Schnittstellen benutzen, die bezüglich der oben angeführten Datenfelder bereits 4-Byte Felder verwenden, ihrerseits jedoch explizit oder implizit von der bisherigen Beschränkung auf Werte kleiner `X'00FFFFFF'` Gebrauch machen.

Im folgenden werden einige Problempunkte beispielhaft aufgeführt:

- Die höchste 3-Byte-Blocknummer `X'FFFFFF'` hat eine spezielle Bedeutung.
- „Blocknummern“ > `X'00FFFFFF'` repräsentieren nicht Blöcke, sondern andere Objekte.
- Bei Berechnungen mit Blocknummern oder Blockzählern > `X'00FFFFFF'` kann es zum Überlauf kommen.
- Die Stellenzahl von Ein- oder Ausgabefeldern reicht nicht zur Darstellung beliebig großer Blocknummern oder Blockzähler.
- Bei Umrechnungen von Hexadezimalzahlen in Dezimalzahlen ist die Feldlänge für die Dezimalzahl zu klein.
- Es wird unterstellt, dass Datenstrukturen, deren Umfang von einer Dateigröße abhängt, stets im virtuellen Speicher Platz finden. Diese Annahme kann für Dateien < 32 GB gültig sein, nicht aber, wenn diese Größe überschritten wird.

Ausführliche und zusammenhängende Informationen zum Thema „Große Dateien“ finden Sie im Handbuch „Dateien und Volumes größer 32 GB“ [18].

11.4 XS-/Nicht-XS-Schnittstelle

Neben der XS-Schnittstelle (31-Bit-Adressierung) wird kompatibel die Nicht-XS-Schnittstelle mit 24-Bit-Adressierung unterstützt. Das heißt, Programme mit 24-Bit-Adressierung sind – je nach Voreinstellung des Assemblers – ohne Umstellung ablauffähig.
Der Benutzer profitiert dabei von dem auf 16 MByte erweiterten Benutzeradressraum. Umgekehrt sind Programme, die mit 31-Bit-Adressen arbeiten, voll im unteren Adressraumbereich ablauffähig. Programme mit 24-Bit-Adressierung können auch mit XS-fähigen Programmen verbunden werden.

11.4.1 FCB-Behandlung beim OPEN

Arbeitet ein Programm mit der 24-Bit-Schnittstelle, erfolgt die Dateiverarbeitung über einen 24-Bit-TU-FCB. Im Gegensatz zum 31-Bit-TU-FCB (XS) wird im 24-Bit-TU-FCB Platz freigehalten für die logischen Routinen, die in den TU-FCB kopiert werden. Mit dem Operanden FORM=SHORT im FCB-Makroaufruf kann das Laden der logischen Routinen unterdrückt werden, der Benutzer muss dann allerdings selbst das Blocken und Entblocken der Datensätze durchführen. Bei normaler Verarbeitung von SAM- und ISAM-Dateien kann die Datei in Zusammenhang mit FORM=SHORT nicht eröffnet werden.

11.4.2 Generierungsmodus

Die Wahl der DVS-Schnittstelle wird durch den Generierungsmodus gesteuert. Er ist voreingestellt durch den Assembler, kann aber auch global in einem Programm mit dem Makroaufruf GPARMOD eingestellt werden oder in jedem einzelnen Makroaufruf mit dem Operanden PARMOD.

Innerhalb eines Programms können zwar für verschiedene Dateien verschiedene Schnittstellen gewählt werden, aber alle Makroaufrufe für eine Datei/einen FCB (OPEN, CLOSE, EXLST, FCB, Aktionsmakros) müssen mit demselben Generierungsmodus übersetzt werden.

Beispiel

Bei Shared-Update-Verarbeitung können verschiedene Programme die gleiche Datei in unterschiedlichem Generierungsmodus bearbeiten. Bei Kettung von PAM-Makroaufrufen können innerhalb der Kette verschiedene Generierungsmodi gewählt werden.

```
START
             USING *,3
             LDBASE 3,ORG=YES
             :
             OPEN   FCB1,PARMOD=24
             OPEN   FCB2,PARMOD=31
             :
             GET    FCB1,PARMOD=24
             PUT    FCB2,PARMOD=31
             :
             CLOSE ALL,PARMOD=31
             TERM
             :
   FCB1      FCB    PARMOD=24,EXIT=EXLST1
   EXLST1    EXLST  PARMOD=24
             :
   FCB2      FCB    PARMOD=31,EXIT=EXLST2
   EXLST2    EXLST  PARMOD=31
             :
             END
```

Beispiel

Kettung von PAM-Makroaufrufen mit verschiedenem Generierungsmodus

```
ELEM1    PAM    FCB2,CHAIN=ELEM2,PARMOD=31
ELEM2    PAM    FCB1,CHAIN=ELEM3,PARMOD=24
ELEM3    PAM    FCB2,PARMOD=31
```

11.4.3 Adressierungsmodus

Alte, bestehende Objektmoduln und Programme, die mit dem alten Generierungsmodus übersetzt wurden (entspricht PARMOD=24), sind weiterhin ablauffähig, allerdings nur im 24-Bit-Adressraum, d.h. die Adressierung ist 24-Bit-abhängig.

Programme, die mit dem Generierungsmodus für 31-Bit-Adressierung übersetzt wurden, sind sowohl im 24-Bit- als auch im 31-Bit-Adressierungsmodus ablauffähig, sie sind also adressierungsmodus-unabhängig.

Auch hier gilt, dass innerhalb eines Programms eine Datei im 24-Bit-Adressierungsmodus und eine weitere im 31-Bit-Adressierungsmodus oder gemischt bearbeitet werden können. Es muss aber sichergestellt werden, dass vor einem DVS-Makroaufruf der für den FCB gültige Adressierungsmodus eingestellt ist.

11.4.4 Umstellungshinweise 24-/31-Bit

Die PARMOD=31-Expansionen der DVS-Makros unterscheiden sich i.a. von den PARMOD=24-Expansionen, was in Einzelfällen zum Überlaufen von Basisregister u.Ä. führen kann. Die PARMOD=31-Expansionen beginnen alle mit einem 8 Byte langen Standardheader. Dieser Standardheader ist im Anhang des Handbuches „DVS-Makros“ [1] beschrieben.

Zugriffsmethoden SAM und ISAM

Die Aktionsmakros, die mit der 31-Bit-Schnittstelle generiert werden, zerstören immer das Register 15. Auch die Register 0, 1 und 14 können zerstört werden, z.B. in Fehlerfällen.

Die von SAM unterstützte Wiedergewinnungsadresse ist für die 31-Bit-Schnittstelle den FCB-Feldern ID1BLK# (Blockzähler) und ID1REC# (Satzzähler) zu entnehmen; im 24-Bit-TU-FCB ist sie im Feld ID1RPTR enthalten (siehe Abschnitt „Wiedergewinnungsadresse“).

12 Kennsatz-Verarbeitung beim Eröffnen von Banddateien

Im Verlauf der OPEN-Verarbeitung werden (abhängig vom Kennsatzaufbau der Datei) Kennsatzprüfungen durchgeführt, und zwar in folgender Reihenfolge:

1. Bandanfangs-Kennsätze
2. Dateianfangs-Kennsätze

Verläuft eine Prüfung nicht erfolgreich, gibt das DVS eine Meldung aus, die auf den betreffenden Fehler hinweist und dem Operator oder dem Benutzer die Möglichkeit bietet, auf den Fehler zu reagieren. Solche Reaktionen können sein:

- Wiederholung der Kennsatzprüfung
- Anstoß einer Exit-Routine im Programm
- Programmabbruch

Ob Meldungen ausgegeben oder unterdrückt werden, hängt ab von der TAPE-ACCESS-Anzeige im Benutzerkatalog und vom SECLEV-Operanden des FILE-/FCB-Makroaufrufs bzw. vom Operanden PROTECTION-LEVEL im Kommando ADD-FILE-LINK.

Der Ablauf der Kennsatz-Verarbeitung variiert in geringem Ausmaß mit der Angabe im LABEL- bzw. LABEL-TYPE-Operanden im Kommando- oder Makroaufruf, sowohl bei Eingabe- als auch bei Ausgabedateien.
So legt der Benutzer mit der Angabe LABEL=STD oder LABEL=(STD,n) im Makro bzw. mit LABEL-TYPE=*STD bzw. LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n) im Kommando fest, dass er eine Datei mit Standardkennsätzen verarbeiten will.
Die Angabe LABEL=NSTD bzw. LABEL-TYPE=*NON-STD wird dahingehend interpretiert, dass die zu verarbeitende Datei Nicht-Standardkennsätze haben wird, entsprechend verlaufen die Kennsatzprüfungen durch das DVS.
Das Gleiche gilt für die Angabe LABEL=NO bzw. LABEL-TYPE=*NO: das DVS „erwartet“ eine Datei ohne Kennsätze.

In den folgenden Abschnitten wird die Kennsatzbehandlung beim OPEN wie folgt beschrieben: zunächst die Kennsatz-Verarbeitung für Eingabedateien, dann die Kennsatz-Verarbeitung für Ausgabedateien, jeweils in der Reihenfolge: Datei mit Standardkennsätzen, Datei mit Nicht-Standardkennsätzen, Datei ohne Kennsätze.

Der Aufbau der Routinen, in denen Benutzer-Kennsätze/Nicht-Standardkennsätze verarbeitet werden, ist im Abschnitt „EXLST-Ausgänge der Kennsatz-Verarbeitung für Banddateien“ beschrieben.

12.1 Voraussetzungen für die Kennsatz-Verarbeitung

- TAPE-ACCESS-Berechtigung (TPIGNORE)
- BYPASS-Behandlung
- Eigentümerkennzeichen
- DVS-Meldungen der Kennsatz-Verarbeitung
- Wiederholung der Kennsatzprüfung (RETRY)

12.1.1 TAPE-ACCESS-Berechtigung (TPIGNORE)

„TPIGNORE“ steht für „Ignore Error on Tape“ und bezieht sich auf die Berechtigung des Benutzers (bzw. der Benutzerkennung), Fehlermeldungen der Kennsatz-Verarbeitung zu ignorieren.

Der Dateieigentümer kann – wenn er im HDR3-Kennsatz als Eigentümer eingetragen ist – alle entsprechenden Fehlermeldungen ignorieren oder auf die Kennsatzprüfung verzichten (Operand SECLEV im FILE-/FCB-Makroaufruf; Operand PROTECTION-LEVEL im Kommando ADD-FILE-LINK).
Will der Benutzer auf „fremde“ Banddateien zugreifen, kann er Fehlermeldungen nur dann ignorieren oder Kennsatzprüfungen umgehen, wenn ihm die Systembetreuung die entsprechende TAPE-ACCESS-Berechtigung erteilt hat. Diese Berechtigung ist im Benutzerkatalog im Feld TAPE-ACCESS vermerkt. Informationen über die Einträge im Benutzerkatalog für die eigene Benutzerkennung erhält der Benutzer mit dem Kommando SHOW-USER-ATTRIBUTES.

Die folgende Tabelle erläutert die Stufen der TAPE-ACCESS-Berechtigung.

TAPE-ACCESS-Wert	**Bedeutung**
STD	Der Benutzer darf Fehlermeldungen nicht ignorieren.
PRIVILEGED	Der Benutzer darf Fehlermeldungen ignorieren; im Batchbetrieb unterdrückt das DVS bestimmte Meldungen, wenn PROTECTION-LEVEL=LOW gilt; bei PROTECTION-LEVEL=HIGH muss der Operator entscheiden, ob der Fehler ignoriert werden darf (Kommando ADD-FILE-LINK, Operand PROTECTION-LEVEL).
READ	Der Benutzer darf Fehlermeldungen ignorieren, die sich auf Eingabedateienbeziehen; die Kennsatzprüfung wird nicht ausgeschaltet.
BYPASS-LABEL	Der Benutzer darf die Kennsatzprüfung umgehen und/oder Fehlermeldungenfür Eingabedateien ignorieren.
ALL	Der Benutzer darf die Kennsatzprüfung umgehen und/oder alle Fehlermeldungenignorieren.

Tabelle 30: TAPE-ACCESS-Berechtigung

Meldungen, die das Montieren der Bänder oder die Bandkennsatzprüfungen betreffen, gehen an den Operator. Der Operator entscheidet über den weiteren Verlauf der Bandverarbeitung. Meldungen über Dateikennsatzprüfungen gehen an den Benutzer bzw. an den aufrufenden Auftrag. Der Benutzer kann in seinem Programm entsprechende Fehlerroutinen aktivieren.

Die TAPE-ACCESS-Berechtigung wirkt sich auch auf die BYPASS-Behandlung aus.

12.1.2 BYPASS-Behandlung

Ist beim OPEN für eine Band-Eingabedatei (OPEN INPUT/REVERSE) im FILE-Aufruf BYPASS=LP bzw. im Kommando ADD-FILE-LINK der Operand BYPASS-LABEL-CHECK angegeben und hat der Benutzer die entsprechende Berechtigung, werden alle Kennsatzprüfungen übergangen. Das Band wird entsprechend der zweiten Angabe im BYPASS-Operanden bzw. entsprechend des Wertes für den Operanden BYPASS-LABEL-CHECK positioniert.

Fehlt die Berechtigung, wird der OPEN mit Fehlermeldung abgebrochen.

Die Montage des Bandes wird entsprechend den Benutzerangaben von der Geräteverwaltung überwacht.

Da alle Kennsatzprüfungen übergangen werden, kann das System keine Code-Prüfung übernehmen. Ist das Band in einem anderen als EBCDIC- oder ISO-7-Code beschrieben, muss der Benutzer mit einer eigenen Code-Tabelle arbeiten.

Ist die BYPASS-Funktion zusammen mit LABEL=NSTD im Makro bzw. mit LABEL-TYPE=*NON-STD im Kommando angegeben, wird lediglich die systemseitige Überprüfung der Bandanfangs-Kennsätze übergangen. Die Benutzerroutinen zur Behandlung von Kennsätzen werden wie gewohnt aktiviert. Die Positionierungsangabe im BYPASS-Operanden wird bei LABEL=NSTD bzw. bei LABEL-TYPE=*NON-STD nur ausgewertet, wenn keine UVL-Kennsätze zu prüfen sind.

12.1.3 Eigentümerkennzeichen

Der Bandanfangs-Kennsatz VOL1 enthält ein „Eigentümerkennzeichen“, das für die Überprüfung der Zugriffsberechtigung ausgewertet wird.

Das „Eigentümerkennzeichen“ ist in der Regel nicht mit Inhalt belegt oder gibt als Eigentümer des Bandes das Rechenzentrum an. In diesen Fällen darf jeder Benutzer auf das Band zugreifen.

Verweist das „Eigentümerkennzeichen“ auf einen Bandeigentümer, wird beim Erstellen der ersten Datei auf dem Band der „Zugriffsvermerk“ aus dem HDR1-Kennsatz übernommen, wenn die Datei vom Bandeigentümer erstellt wird. Ist eine Einschränkung im Zugriffsvermerk vorhanden, kann nur der Bandeigentümer auf das Band zugreifen. Sind Bandeigentümer und Eigentümerkennung gleich oder ist die Bandeigentümerkennzeichnung leer, wird der Zugriffsvermerk als uneingeschränkt angenommen.

12.1.4 DVS-Meldungen der Kennsatz-Verarbeitung

Während der Kennsatz-Verarbeitung kann das DVS mit folgenden Meldungen auf Widersprüche bei der Kennsatzprüfung reagieren:

```
DMS0DA1  FALSCHE VSN, GERAET (&00). VSN (&01) FUER DATEI (&02) VSEQ (&03) EINHAENGEN.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=HDR DRUCKEN; I=IGNOR.)
DMS0DA2  KEIN VOL1/HDR AUF BAND: GER. (&00). VSN (&01) FUER DATEI (&02)
           VSEQ (&03) EINHAENGEN.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=HDR DRUCKEN)
DMS0DA3  BAND SCHREIBGESCHUETZT: VSN '(&00)' FUER DATEI '(&01)' VSEQ '(&02)'.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=HDR DRUCKEN; I=IGNOR.)
DMS0DA5  FALSCHE DATEI-IDENTIFIKATION: VSN '(&00)' FUER DATEI '(&01)'
           VSEQ '(&02)'.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=HDR DRUCKEN; I=IGNOR.)
DMS0DD6  FALSCHES DATEIMENGEN-KENNZEICHEN: VSN '(&00)' FUER DATEI '(&01)'
           VSEQ '(&02)'.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=KENNS.DRUCK; I=IGNOR.)
DMS0DD7  BENUTZER DARF AUF DAS BAND NICHT ZUGREIFEN. VSN '(&00)' FUER DATEI
           '(&01)' VSEQ '(&02)'.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERHOLUNG)
DMS0DD8  KEINE STANDARD-KENNSAETZE AUF BAND MIT VSN '(&00)' FUER DATEI
           '(&01)' VSEQ '(&02)'.
           ANTWORT (0=FEHLERAUSG.; 1=WIEDERH.; 2=KENNSATZ DRUCK)
DMS0DF6  EINGABEBAND MIT ARCHIVNUMMER (&00) HAT STD-KENNSAETZE, SOLL ABER
           MIT 'NSTD' ODER 'NO'
           VERARBEITET WERDEN. ANTWORT (0=FEHLERAUSG.; C=FORTS.)
DMS0DFB  VSN '(&03)' AUF '(&00)' FUER DATEI '(&01)' VSEQ '(&02)' I. O.?
           ANTWORT (TSN.0=FEHLERAUSG.;
           TSN.1=WIEDERH.; TSN.2'XXXXXX'=NEUE VSN; TSN.=AKZEPT.)
```

Dabei bedeuten:

`FEHLERAUSGANG`	im Programm wird – falls vorhanden – eine Exit-Routine angestoßen, die diesen Fehler abfängt und das Programm ordnungsgemäß beendet
`WIEDERHOLUNG`	die Kennsatzprüfung wird wiederholt (siehe Abschnitt „Wiederholung derKennsatzprüfung (RETRY)“)
`HDR DRUCKEN`	der Inhalt des geprüften HDR-Kennsatzes wird über die Systemdatei SYSOUT am Bildschirm ausgegeben bzw. bei Batchverarbeitung in die SYSOUT-Datei geschrieben
`NEUE VSN`	nach einem Bandwechsel soll die Kennsatzprüfung für das neue Band wiederholt werden (siehe Abschnitt „Wiederholung der Kennsatzprüfung (RETRY)“ (Wiederholung der Kennsatzprüfung (RETRY))
`IGNORIEREN`	abhängig von der TAPE-ACCESS-Berechtigung und dem Wert des SECLEV-Operanden im FILE-/FCB-Aufruf bzw. des Operanden PROTECTION-LEVEL im Kommando ADD-FILE-LINK kann der aufgetretene Fehler ignoriert und der Programmlauf fortgesetzt werden
`TSN`	Auftragsnummer des laufenden Auftrags, für den die Kennsatzprüfung durchgeführt wurde

VSN	Archivnummer des Bandes, mit dem die Kennsatzprüfung durchgeführt wurde, bzw. Archivnummer des neu einzuhängenden Bandes

12.1.5 Wiederholung der Kennsatzprüfung (RETRY)

Normalerweise ist ein Bandgerät bereits mit dem richtigen Datenträger belegt, wenn im FILE-Aufruf der VOLUME-Operand bzw. in den Kommandos CREATE-FILE, IMPORT-FILE oder CREATE-TAPE-SET der Operand VOLUME korrekt versorgt wurde.

Dennoch kann es in Ausnahmefällen vorkommen, dass der Operator ein „falsches“ Band einhängt. Bei der Kennsatzprüfung wird dann die Aufforderung „RETRY WITH ANOTHER VOLUME“ ausgegeben.

Der Benutzer kann jetzt im Dialog die Verarbeitung wiederholen. Tritt der Fehler auch bei der Wiederholung auf, wird der Benutzer aufgefordert, eine neue Archivnummer anzugeben, damit ein anderes Band eingehängt wird. Bei Ausgabe- und FOREIGN-Dateien wird der Katalogeintrag mit dieser neuen Archivnummer aktualisiert, sobald der Montiervorgang abgeschlossen ist.

Bevor der Operator die Fehlermeldung an der Konsole zur erneuten Bandverarbeitung beantwortet, kann er das Band im Fehlerfall auswechseln. Die Archivnummer des neuen Bandes ist normalerweise mit der im Katalog eingetragenen VSN identisch.

Unterscheidet sich die VSN des neuen Bandes von der angeforderten, bewirkt dies eine Änderung des Katalogeintrags. Dies trifft in folgenden Fällen zu:

- Die Datei wurde mit oder ohne TSET-Angabe erstellt (OPEN OUTPUT/OUTIN) und wird erstmals oder bei Bandwechsel benutzt
- Die Datei wird importiert (STATE=FOREIGN bzw. Kommando IMPORT-FILE)
- Die Datei wird über die schon voll belegten Bänder hinaus erweitert

In diesen Fällen wird eine „Acknowledge-VSN“ an der Konsole ausgegeben, sobald der Operator die RETRY-Meldung beantwortet hat.

12.2 Kennsatz-Verarbeitung für Eingabedateien

Der erste Schritt einer Kennsatzverarbeitung ist in der Regel die Prüfung der Bandanfangs-Kennsätze. Dieser Schritt entfällt, wenn für das Band bereits vorher eine erfolgreiche VOL1-Prüfung durchgeführt wurde. In diesem Fall wird das Band sofort auf Dateianfang positioniert und die Kennsatzprüfung mit den HDR-Kennsätzen fortgesetzt.

12.2.1 Datei mit Standardkennsätzen

Die Verarbeitung von Eingabedateien mit Standardkennsätzen wird im Makro FILE mit dem Operanden LABEL=(STD,n), im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n) angefordert.

Bandanfangs-Kennsätze prüfen

1. Ist der erste Satz auf dem Band der Bandanfangs-Kennsatz VOL1?

Der Bandanfangs-Kennsatz wird identifiziert über die Zeichenfolge VOL1 in den Feldern Kennsatzname und -nummer (das sind die ersten vier Zeichen auf einem Band mit Standardkennsätzen). Enthält das Band diese Zeichenfolge nicht, d.h. ist der erste Satz kein VOL1-Kennsatz, gibt das DVS die Meldung `DMS0DA2` aus, auf die der Operator reagieren muss.

2. Ist die Datei bereits katalogisiert oder handelt es sich um eine FOREIGN-Datei bzw. eine zu importierende Datei? Hat der Benutzer für die FOREIGN-Datei bzw. für die zu importierende Datei bereits ein Band angefordert?

Ist die Datei bereits katalogisiert, sind die Archivnummern der von ihr belegten Bänder im Katalogeintrag enthalten. Für FOREIGN-Dateien (d.h. zu importierende, noch nicht katalogisierte Dateien) müssen Sie die Archivnummern der benötigten Bänder im VOLUME-Operanden des FILE-Aufrufs bzw. im Operanden VOLUME der Kommandos IMPORT-FILE bzw. CREATE-FILE benennen. Haben Sie dies unterlassen, gibt das DVS zum Zeitpunkt der OPEN-Verarbeitung an der Konsole die Meldung `DMS0DFB` aus und gibt so dem Operator die Möglichkeit, das montierte Band zu überprüfen.

3. Stimmt die Archivnummer (VSN) im VOL1-Kennsatz mit der vom Benutzer im FILE-Aufruf bzw. in den Kommandos IMPORT-FILE bzw. CREATE-FILE angegebenen überein?

Das DVS vergleicht den Inhalt des dritten VOL1-Kennsatzfeldes (VSN des Bandes), mit der vom Benutzer angegebenen VSN. Stimmen diese beiden nicht überein, gibt das DVS an der Konsole die Meldung `DMS0DA1` aus. Der Operator kann jetzt den Kennsatz überprüfen, ein neues Band einhängen, das Programm abbrechen (OPENER-Routine) oder die Meldung ignorieren (wenn die entsprechende TPIGNORE-Berechtigung vorliegt).

4. Ist Zugriff nur dem Bandeigentümer gestattet? Wenn ja: Sind Bandeigentümer- und Benutzerkennung identisch?

Das DVS wertet den Inhalt der beiden VOL1-Felder „Zugriffsvermerk“ und „Eigentümerkennzeichen“ aus. Ist der Benutzer nicht zugriffsberechtigt, gibt das DVS dem Benutzer die Meldung `DMS0DD7` aus. Sie können jetzt Ihr Programm beenden oder die Kennsatzprüfung wiederholen lassen, für FOREIGN-Dateien mit einem anderen Band.

5. Enthält ein Nicht-BS20000-Band weitere VOL-Kennsätze?

Das DVS ignoriert die Bandanfangs-Kennsätze VOL2 bis VOL9, d.h. es wertet sie nicht aus, sondern setzt die OPEN-Verarbeitung mit dem nächsten Schritt fort.

6. Folgen den Standard- noch Benutzer-Bandanfangs-Kennsätze (UVL)?

UVL-Kennsätze müssen in Routinen, die über den OPENV-Ausgang des EXLST-Makroaufrufs angesprungen werden, vom Programm selbst ausgewertet werden. Enthält das Band UVL-Kennsätze (UVL1 bis UVL9), aktiviert das DVS für jeden Kennsatz die OPENV-Routine; die Adresse des Kennsatzes ist im Register 0 enthalten. Die Kennsatzprüfung wird jeweils mit dem LBRET-Makroaufruf abgeschlossen. Enthält das Programm keine OPENV-Routine zur UVL-Auswertung, ignoriert das DVS diese Kennsätze.

Positionieren

Als nächsten Schritt positioniert das DVS auf den Anfang der einzulesenden Datei. Die Dateiposition auf dem Band entnimmt es dem FILE-/FCB-Makroaufruf (Operand FSEQ), dem Kommando ADD-FILE-LINK (Operand FILE-SEQUENCE) oder dem Katalogeintrag (Feld FSEQ). Ist die Datei eine FOREIGN-Datei, deren Bandposition unbekannt ist, können Sie im Makro FILE/FCB FSEQ=UNK (= unknown) bzw. im Kommando ADD-FILE-LINK FILE-SEQUENCE=*UNKNOWN angeben. In diesem Fall positioniert das DVS hinter die Bandanfangs-Kennsätze und durchsucht das gesamte Band. Wenn das Band richtig positioniert ist, prüft das DVS die Dateianfangs-Kennsätze HDR1 bis HDR3 und UHL0 bis UHL255.

Dateianfangs-Kennsätze prüfen

1. Ist der erste Satz der Datei bzw. der erste auf die Bandanfangs-Kennsätze folgende Satz der HDR1-Kennsatz?

Der HDR1-Kennsatz beginnt mit der Zeichenfolge HDR1 in den ersten vier Bytes.
Findet das DVS diese Zeichenfolge nicht, gibt es dem Benutzer die Meldung `DMS0DD8` aus. Sie können nach einer Korrektur die Kennsatzprüfung wiederholen lassen. Wollten Sie eine FOREIGN-Datei lesen, müssen Sie diese auf einem anderen Band suchen.

Findet das DVS einen HDR1-Kennsatz, liest es anschließend den HDR2-Kennsatz. Den HDR3-Kennsatz liest es nur dann, wenn es am System-Code des HDR1-Kennsatzes erkannt hat, dass die Banddatei im BS2000 erstellt worden ist. Nur in diesem Fall werden auch die unter 2 und 3 beschriebenen Schritte ausgeführt.

2. Stimmen Dateiname in HDR1- und HDR3-Kennsatz und der vom Benutzer angegebene Dateiname überein?

Das DVS vergleicht den Dateinamen in HDR1- und HDR3-Kennsatz mit dem Dateinamen aus dem Makro FILE bzw. aus den Kommandos IMPORT-FILE oder CREATE-FILE. Stimmen sie nicht überein, gibt das DVS dem Benutzer die Meldung `DMS0DA5` aus, auf die er mit „Programmbeendigung“, „Wiederholung der Kennsatzprüfung“, „Kennsatz ausgeben“ oder „Ignorieren“ (abhängig von der TPIGNORE-Berechtigung) reagieren kann.

3. Ist der Dateizugriff nur dem Dateieigentümer vorbehalten? Wenn ja: Stimmen Dateieigentümerkennzeichen im HDR3-Kennsatz und Benutzerkennung des Auftrags überein?

Das HDR1-Feld „Zugriffsvermerk“ zeigt an, ob die Datei mehrbenutzbar ist.
Ist die Datei nicht mehrbenutzbar, vergleicht das DVS die im HDR3-Kennsatz eingetragene Eigentümerkennung mit der Benutzerkennung des Auftrags. Stimmen diese beiden nicht überein, gibt das DVS dem Benutzer die Meldung `DMS0DD7` aus, auf die er wie bei der VOL1-Prüfung reagieren kann: Programmbeendigung oder Wiederholung der Kennsatzprüfung, bei FOREIGN-Dateien mit einem anderen Band.

4. Stimmt die im HDR1-Kennsatz eingetragene Dateiabschnittsnummer mit der im FILE bzw. ADD-FILE-LINK-Aufruf angegebenen überein?

Die Dateiabschnittsnummer, die im HDR1-Kennsatz eingetragen ist, gibt die Position eines Dateiabschnitts innerhalb eines Volume Sets (einer Bandmenge) an. Mit dem VSEQ-Operanden des FILE-Makroaufrufs bzw. mit dem Operanden VOL-SEQUENCE-NUMBER im Kommando ADD-FILE-LINK haben Sie festgelegt, welchen Dateiabschnitt einer „Mehrbanddatei“ Sie bearbeiten wollen. Stimmen diese beiden Angaben nicht überein, gibt das DVS dem Benutzer die Meldung `DMS0DD6` aus. Sie können diese Meldung – falls Sie dazu berechtigt sind – ignorieren oder in sonst geeigneter Weise darauf reagieren.

5. Sind Kennwörter in den Katalogeintrag zu übernehmen?

Für FOREIGN-Dateien, die über FILE-Makroaufruf bzw. über das Kommando IMPORT-FILE katalogisiert werden, werden die im HDR3-Kennsatz enthaltenen Kennwörter in den neuen Katalogeintrag übernommen.

6. Ist die Datei durch eine Schutzfrist oder mit Schreibschutz gesperrt?

Wenn die Datei mit OPEN INOUT oder OPEN EXTEND (siehe OPEN-Modi, Handbuch „DVS-Makros“ [1]) eröffnet werden soll, prüft das DVS das HDR3-Feld „Zugriffsart“, das der ACCESS-Angabe im Katalogeintrag entspricht. Im HDR1-Kennsatz wird das Feld „Freigabedatum“ überprüft, das anzeigt, ob für die Datei noch eine Schreibschutzfrist besteht oder ob sie überschrieben werden kann. Eine Schutzfrist kann mit dem Operanden RETPD in FILE-/FCB-Makroaufruf bzw. mit dem Operanden RETENTION-PERIOD im Kommando ADD-FILE-LINK bei der Dateierstellung vereinbart worden sein.

Besteht über „Zugriffsart“ oder „Freigabedatum“ noch ein Schreibschutz, gibt das DVS dem Benutzer die Meldung `DMS0DA3` aus. Sie können dann – falls Sie dazu berechtigt sind – diese Meldung ignorieren und in der Dateiverarbeitung fortfahren oder die Kennsatzprüfung wiederholen lassen (bei FOREIGN-Dateien mit neuem Band) bzw. Ihr Programm über eine Exit-Routine (OPENER) ordnungsgemäß beenden lassen.

7. Folgen den HDR-Kennsätzen noch Benutzer-Kennsätze (UHL)?

Benutzer-Dateianfangs-Kennsätze (UHL0 bis UHL255) müssen vom Programm in einer Exit-Routine ausgewertet werden: LABGN-Ausgang des EXLST-Makroaufrufs.
Das DVS aktiviert für jeden UHL-Kennsatz die LABGN-Routine; die Kennsatzadresse ist im Register 0 enthalten. Die Kennsatzprüfung wird jeweils mit dem LBRET-Makroaufruf abgeschlossen. Ist im Programm keine LABGN-Routine vorgesehen, übergeht das DVS die UHL-Kennsätze.

12.2.2 Datei mit Nicht-Standardkennsätzen

Die Verarbeitung von Eingabedateien mit Nicht-Standardkennsätzen wird im Makro FILE mit dem Operanden LABEL=NSTD, im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE=*NON-STD angefordert.

Kennsätze prüfen

Zunächst prüft das DVS, ob das Band mit einem VOL1-Kennsatz beginnt. Ist das nicht der Fall, wird der Dateizugriff ohne weitere Prüfungen zugelassen (es geht weiter mit Punkt 2, "Datei mit Standardkennsätzen").

Enthält das Band einen VOL1-Kennsatz, wird wie bei Dateien mit Standardkennsätzen die Gleichheit der Archivnummern im Makro FILE bzw. im Kommando ADD-FILE-LINK und Kennsatz überprüft. Stimmen sie nicht überein, gibt das DVS dem Operator die Meldung `DMS0DA1` aus bzw. bei FOREIGN-Dateien, für die im Makro FILE bzw. im Kommando ADD-FILE-LINK keine Archivnummer angegeben wurde, die Meldung `DMS0DFB`.

Stimmen die Archivnummern überein, hängt es vom Inhalt der HDR-Kennsätze ab, ob der Dateizugriff zulässig ist oder nicht. Wie bei Dateien, für die LABEL=(STD,n) bzw. LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n) gilt, wird zunächst der HDR1-Kennsatz gelesen, anschließend HDR2- und HDR3-Kennsatz.

Es wird der HDR1-„Zugriffsvermerk“ geprüft und für „nicht-mehrbenutzbare“ Dateien das Dateieigentümerkennzeichen mit der Benutzerkennung des Auftrags verglichen (Fehlermeldung `DMS0DD7`). Soll die Datei mit OPEN INOUT eröffnet werden, werden anschließend HDR3-„Zugriffsart“ (ACCESS=READ) und HDR1-„Freigabedatum“ überprüft (Fehlermeldung `DMS0DA3`).

Kennsätze auswerten

Zunächst wird das Band zur Bandanfangsmarke zurückgespult. Danach verzweigt das DVS zur LABGN-Exit-Routine des Benutzerprogramms, in der die Bandanfangs-Kennsätze ausgewertet werden sollen. Die Auswertung der Kennsätze erfolgt über BTAM-Makroaufrufe.

Ist im EXLST-Makroaufruf kein LABGN-Ausgang definiert, werden die Kennsätze ignoriert.

Werden die Kennsätze nicht gültig ausgewertet, wird das Programm mit einem OPEN-Fehler beendet.

Die LABGN-Routine wird mit dem LBRET-Makroaufruf abgeschlossen.

Positionieren

Sie sind für die korrekte Positionierung selbst verantwortlich, d.h. Sie müssen dafür sorgen, dass das Band auf die Dateianfangs-Kennsätze positioniert wird, bevor Sie die Ablaufsteuerung an das DVS zurückgeben. Auch die Dateianfangs-Kennsätze müssen vom Programm ausgewertet werden. Sie sind auch für deren korrekte Positionierung verantwortlich.

Tape Mark

Bandabschnittsmarken (Tape Mark) werden nur berücksichtigt, wenn TPMARK=YES im Dateisteuerblock oder im TFT-Eintrag gilt; das DVS positioniert hinter die Abschnittsmarke. Gilt TPMARK=NO im Dateisteuerblock oder im TFT-Eintrag, nimmt das DVS an, dass der Benutzer das Band bereits auf den ersten Datenblock positioniert hat (siehe oben).

12.2.3 Datei ohne Kennsätze

Die Verarbeitung von Eingabedateien ohne Kennsätze wird im Makro FILE mit dem Operanden LABEL=NO, im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE= *NO angefordert.

Unabhängig von der Benutzerangabe zu LABEL bzw. LABEL-TYPE wird der Bandanfangs-Kennsatz geprüft – falls noch keine VOL1-Prüfung durchgeführt wurde.

Kennsätze prüfen

Zunächst prüft das DVS, ob das Band mit einem VOL1-Kennsatz beginnt. Falls nein, kann der Auftrag ohne weitere Überprüfungen auf die Datei zugreifen (siehe Abschnitt „Datei ohne Kennsätze“, unten).
Beginnt das Band trotz der Angabe LABEL=NO bzw. LABEL-TYPE=*NO mit einem VOL1-Kennsatz, wird zuerst die Gültigkeit der Archivnummer geprüft. Wie bei Dateien mit Standardkennsätzen gibt das DVS im Fehlerfall dem Operator die Meldung `DMS0DA1` aus bzw. bei FOREIGN-Dateien die Meldung `DMS0DFB`.

Im nächsten Schritt prüft das DVS über VOL1-„Zugriffsvermerk“ und -„Bandeigentümerkennzeichen“ die Zugriffsberechtigung der Benutzerkennung (Fehlermeldung `DMS0DD7`). Anschließend liest das DVS die HDR-Kennsätze und überprüft die Zugriffsberechtigung anhand von HDR1-Zugriffsvermerk und HDR3-Dateieigentümerkennzeichen (Fehlermeldung `DMS0DD7`). Soll die Datei mit OPEN INOUT/EXTEND eröffnet werden, werden noch HDR3-Zugriffsart und HDR1-Freigabedatum überprüft (Fehlermeldung `DMS0DA3`).
Im Dialogbetrieb gibt das DVS die Fehlermeldung `DMS0DF6` aus: Sie werden darauf aufmerksam gemacht, dass die LABEL-Angabe nicht mit dem Bandinhalt übereinstimmt. Sie können entscheiden, ob Sie die Datei dennoch verarbeiten wollen, d.h. ob der Programmlauf fortgesetzt oder mit einem OPEN-Fehler beendet werden soll.

Positionieren

Ist der erste Satz kein VOL1-Kennsatz, sondern eine Abschnittsmarke (Tape Mark) und gilt TPMARK=YES (vgl. FILE-Makroaufruf/FCB) bzw. TAPE-MARK-WRITE=*YES (siehe Kommando ADD-FILE-LINK), positioniert das DVS hinter diese Abschnittsmarke. Innerhalb des Bandes positioniert das DVS entsprechend den FSEQ-Angaben in Katalogeintrag, FILE oder FCB bzw. entsprechend den Angaben beim Operanden FILE-SEQUENCE im Kommando ADD-FILE-LINK.

12.3 Kennsatz-Verarbeitung für Ausgabedateien

Wie für Eingabedateien müssen auch für Ausgabedateien Band- und Dateianfangs-Kennsätze geprüft werden: ist die neue Datei die erste Datei auf einem noch nicht beschriebenen Band, stammen Band- und Dateianfangs-Kennsätze (VOL1, HDR1, HDR2) aus der Initialisierung des Bandes.

Die Kennsatzverarbeitung hängt – wie bei den Eingabedateien – von der LABEL-Angabe in FILE-Makroaufruf oder FCB ab bzw. vom Operanden LABEL-TYPE im Kommando ADD-FILE-LINK.

12.3.1 Datei mit Standardkennsätzen

Die Verarbeitung von Ausgabedateien mit Standardkennsätzen wird im Makro FILE mit dem Operanden LABEL=(STD,n), im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n) angefordert.

Wie bei Eingabedateien gilt: Wenn der VOL1-Kennsatz bereits überprüft wurde, geht das DVS gleich zur Prüfung der Dateianfangs-Kennsätze über (siehe unter Punkt 5, "Datei mit Standardkennsätzen").

Bandanfangs-Kennsätze prüfen

1. Ist der erste Satz auf dem Band ein Bandanfangs-Kennsatz VOL1?

Das DVS prüft die ersten vier Zeichen auf die Zeichenfolge „VOL1“. Findet es diese Zeichenfolge nicht, ist der erste Satz kein VOL1-Kennsatz, also kein Standardkennsatz. Das DVS gibt an der Konsole die Meldung `DMS0DA2` aus, auf die der Operator reagieren muss.

2. Hat der Benutzer ein bestimmtes Band oder ein Arbeitsband angefordert?

Sie können z.B. im FILE-Makroaufruf mit DEVICE=WORK oder im Kommando CREATE-FILE mit DEVICE-TYPE=WORK ein Arbeitsband anfordern, das Ihnen der Operator zuweisen muss. Der Operator muss auch dann ein Band zuweisen, wenn die Datenträgerliste, die Sie für die Datei per FILE-Makroaufruf (Operand VOLUME) oder Operand VOLUME im Kommando CREATE-FILE definiert haben, erschöpft ist. Das DVS gibt in beiden Fällen an der Konsole die Meldung `DMS0DFB` aus; der Operator überprüft nochmals den MOUNT-Vorgang.

3. Wenn der Benutzer ein bestimmtes Band angefordert hat: Wurde das richtige Band zugewiesen?

Das DVS prüft, ob die im VOL1-Kennsatz enthaltene Archivnummer (VSN) mit der im FILE-Makroaufruf (VOLUME=vsn) bzw. im Kommando CREATE-FILE (Operand VOLUME) angegebenen übereinstimmt. Stimmen diese beiden Archivnummern nicht überein, gibt das DVS an der Konsole die Meldung `DMS0DA1` aus. Der Operator muss jetzt über den weiteren Verlauf der Verarbeitung entscheiden.

4. Ist Zugriff nur dem Bandeigentümer gestattet? Und wenn ja: Sind Bandeigentümer und Benutzerkennung identisch?

Das DVS wertet den Inhalt der beiden VOL1-Felder „Zugriffsvermerk“ und „Eigentümerkennzeichen“ aus. Ist der Benutzer nicht zugriffsberechtigt, gibt das DVS dem Benutzer die Meldung `DMS0DD7` aus. Sie können jetzt Ihr Programm beenden oder die Kennsatzprüfung wiederholen lassen.

In der Regel wird bei der Initialisierung kein Bandeigentümer oder als Bandeigentümer das Rechenzentrum eingetragen und das Band als „mehrbenutzbar“ gekennzeichnet.

5. Ist der nächste zu lesende Satz ein HDR1-Kennsatz?

Im Anschluss an die VOL1-Prüfung wird das Band entsprechend der FSEQ-Angabe im FILE-Makroaufruf oder im FCB bzw. entsprechend der Angabe beim Operanden FILE-SEQUENCE im Kommando ADD-FILE-LINK positioniert.

Findet das DVS an dieser Stelle keinen HDR1-Kennsatz, beginnt es mit den „Maßnahmen gegen implizites Überschreiben“ (siehe Punkt 9, "Datei mit Standardkennsätzen"). Hat das DVS einen HDR1-Kennsatz gefunden, liest es auch den HDR2- und – wenn eine bestehende Datei überschrieben werden soll – auch den HDR3-Kennsatz. Wird ein Band von Beginn an neu beschrieben, ist der nächste Schritt die Aktualisierung der Bandanfangs-Kennsätze (siehe "Datei mit Standardkennsätzen").

6. Darf der anfordernde Auftrag die Datei (über)schreiben?

Das DVS wertet Zugriffsvermerk (im HDR1) und, wenn schon ein HDR3-Kennsatz existiert, Dateieigentümerkennzeichen aus und prüft, ob der Auftrag zugriffsberechtigt ist. Ist der Auftrag nicht zugriffsberechtigt, gibt das DVS dem Benutzer die Meldung `DMS0DD7` aus.

7. Darf die Datei überschrieben werden?

Das DVS prüft Freigabedatum und Zugriffsart in HDR1- und HDR3-Kennsatz. Besteht für die Datei eine Schutzfrist (Freigabedatum in HDR1 > aktuelles Tagesdatum) oder darf laut „Zugriffsart“ in HDR3 die Datei nur gelesen werden (entspricht ACCESS=READ), wird dem Benutzer die Meldung `DMS0DA3` ausgegeben. Sie können, wenn Sie dazu berechtigt sind, diese Meldung ignorieren und mit der Dateiverarbeitung fortfahren. Sie können aber auch

Ihr Programm mit einer Fehlerroutine beenden lassen, sich den Kennsatzinhalt anzeigen lassen oder die Kennsatzprüfung evtl. mit einem neuen Band (bei FOREIGN-Dateien) wiederholen lassen.

Darf die Datei überschrieben werden, werden die Kennwortfelder des HDR3-Kennsatzes nicht mehr geprüft.

> **i** In File-Sets werden die auf eine zu überschreibende Datei folgenden Dateien implizit überschrieben bzw. gelöscht. Ihr Katalogeintrag bleibt zwar bestehen, auf die Daten kann jedoch nicht mehr zugegriffen werden.

8. Haben die Felder „Dateiabschnittsnummer“ und „Dateifolgenummer“ im HDR1-Kennsatz gültige Werte?

Wenn eine bestehende Mehrbanddatei überschrieben wird, muss die Dateiabschnittsnummer im HDR1-Kennsatz mit der im FILE-Makroaufruf (Operand VSEQ=) bzw. mit der im Kommando ADD-FILE-LINK (Operand VOL-SEQUENCE-NUMBER) angegebenen übereinstimmen. Für das Band, auf dem die Datei beginnt, muss VSEQ=1 bzw. VOL-SEQUENCE-NUMBER=1 sein.
Hat VSEQ bzw. VOL-SEQUENCE-NUMBER keinen gültigen Wert, wird die OPEN-Verarbeitung abgebrochen, wenn nicht gleichzeitig (explizit oder implizit) FSEQ=0 oder FSEQ=1 gilt bzw. FILE-SEQUENCE=0 oder FILE-SEQUENCE=1, d.h. wenn die neue Ausgabedatei nicht die erste Datei einer Dateimenge (File Set) ist.

9. Maßnahmen gegen implizites Überschreiben

Wurde in FILE-Makroaufruf oder FCB mit dem Operanden SECLEV bzw. im Kommando ADD-FILE-LINK mit dem Operanden OVERWRITE-PROTECTION=*YES Schutz gegen Überschreiben gefordert (Overwrite Protection (OPR)) und soll die aktuelle Datei als Element einer Dateimenge hinter eine schon bestehende Datei geschrieben werden (FSEQ=n, n > 1), vergleicht das DVS die für die neue Datei definierten Schutzmerkmale „Freigabedatum“ und „Zugriffsart“ mit denen der vorhergehenden Datei (FSEQ=n-1).
Es entnimmt die Werte den entsprechenden Feldern der Kennsätze EOF1 und EOF3.
Hat die vorhergehende Datei keinen EOF3-Kennsatz, gilt implizit ACCESS=WRITE, d.h. Schreibzugriff ist erlaubt. Das Band wird abgewiesen, wenn sowohl für die neue Datei als auch für die vorhergehende Datei eine Schutzfrist vereinbart wurde und das Freigabedatum der neuen Datei höher ist. Das Band wird auch abgewiesen, wenn für die neue Datei ACCESS=READ (nur Lesen erlaubt) gilt, für die Datei mit FSEQ=n-1 jedoch ACCESS= WRITE (Lesen und Schreiben erlaubt).

Bandanfangs-Kennsätze aktualisieren

Wird ein Band ab Bandanfang beschrieben (erste Datei einer Dateimenge oder Folgebandverarbeitung) und sind Datei- und Bandeigentümer identisch, wird der VOL1-Kennsatz aktualisiert in den Feldern „Zugriffsart“ und „Austauschstufe“. Letztere bezieht sich darauf, welche Kennsätze entsprechend der DIN 66029-Austauschstufe standardmäßig unterstützt werden.

Enthält das Benutzerprogramm eine OPENV-Routine zum Schreiben von Benutzer-Band-anfangs-Kennsätzen (UVL), verzweigt das DVS in diese Routine. Ist im EXLST-Makroaufruf kein OPENV-Ausgang definiert, werden keine UVL-Kennsätze geschrieben. Im Register 0 stellt das DVS die Adresse eines 80 Byte großen Bereichs zur Verfügung, in dem die Kennsätze nacheinander erstellt werden können. Die OPENV-Routine wird mit dem LBRET-Makroaufruf abgeschlossen.

Dateianfangs-Kennsätze schreiben/aktualisieren

In die HDR-Kennsätze der neuen Datei werden die im Katalogeintrag hinterlegten Dateischutzmerkmale übernommen und die File-Set-Informationen aus dem FILE-Makroaufruf bzw. aus dem Kommando ADD-FILE-LINK: Dateimengenkennzeichen der ersten Datei, Dateiabschnittsnummer (VSEQ/VOL-SEQUENCE-NUMBER) und Dateifolgenummer (FSEQ/FILE-SEQUENCE).

Enthält das Benutzerprogramm eine LABGN-Routine zum Schreiben von Benutzer-Dateianfangs-Kennsätzen (UHL), verzweigt das DVS in diese Routine. Ist im EXLST-Makroaufruf kein LABGN-Ausgang definiert, werden keine UHL-Kennsätze erzeugt. Verzweigt das DVS in die LABGN-Routine, enthält das Register 0 die Adresse eines 80 Byte großen Bereichs, in dem die UHL-Kennsätze nacheinander erstellt werden können. Die LABGN-Routine wird mit dem LBRET-Makroaufruf abgeschlossen.

Tape Mark schreiben

Standardmäßig schreibt das DVS eine Bandabschnittsmarke (Tape Mark) hinter die Dateianfangs-Kennsätze.

12.3.2 Datei mit Nicht-Standardkennsätzen

Die Verarbeitung von Ausgabedateien mit Nicht-Standardkennsätzen wird im Makro FILE mit dem Operanden LABEL=NSTD, im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE=*NON-STD angefordert.

Im Großen und Ganzen verfährt das DVS bei Dateien, die mit LABEL=NSTD erzeugt werden, genauso wie bei Dateien mit Standardkennsätzen (LABEL=(STD,n)).

Kennsätze prüfen

Ist der erste Satz kein VOL1-Kennsatz, lässt das DVS den Dateizugriff ohne jede weitere Prüfung zu, das Band wird sofort auf Bandanfang positioniert (siehe Punkt 2, "Datei mit Standardkennsätzen").

Ist der erste Satz ein VOL1-Kennsatz, prüft es die darin enthaltene Archivnummer und gibt im Fehlerfall an der Konsole die Meldung `DMS0DA1` aus. Wenn der Auftrag die nötige TPIGNORE-Berechtigung hat, kann der Operator die Meldung ignorieren, ansonsten muss er die geeigneten Maßnahmen ergreifen.

Hat der Benutzer ein Arbeitsband angefordert oder ist die Datenträgerliste der Datei erschöpft, wird der Operator mit der Meldung `DMS0DFB` aufgefordert, ein Band zuzuweisen.

Enthält der VOL1-Kennsatz die korrekte Archivnummer, überprüft das DVS als Nächstes die Dateianfangs-Kennsätze HDR1 bis HDR3. Es wertet die Felder „Zugriffsvermerk“ (HDR1), „Dateieigentümer“ (HDR3), „Freigabedatum“ (HDR1) und „Zugriffsart“ (HDR3) aus. Ist der Auftrag nicht zugriffsberechtigt, erhalten Sie die DMS-Meldung `DMS0DD7` oder `DMS0DA3`. Falls Sie die nötige TPIGNORE-Berechtigung haben, können Sie die Fehlermeldung ignorieren und die Dateiverarbeitung fortsetzen. Ansonsten müssen Sie entsprechend auf die Fehlermeldung reagieren (Programm beenden/Kennsatz ausgeben/Wiederholung).

Positionieren

Ist Dateizugriff zulässig, wird das Band bis zur Bandanfangsmarke zurückgespult.

Kennsätze schreiben

Zunächst verzweigt das DVS in die OPENV-Routine, die die Nicht-Standardkennsätze (UVL) schreibt. Im Register 0 ist die Adresse des Bereichs enthalten, in dem die UVL-Kennsätze nacheinander erstellt werden können. Nicht-Standardkennsätze sind an kein Format gebunden, sondern werden vom Benutzer festgelegt. Die OPEN-Routine wird mit dem EXRTN-Makroaufruf abgeschlossen. Zuvor muss jedoch das Band richtig positioniert werden, bevor die Dateiverarbeitung fortgesetzt werden kann. Enthält das Benutzerprogramm keine OPENV-Routine für UVL-Kennsätze, setzt das DVS voraus, dass das Band bereits richtig positioniert ist.

Tape Mark schreiben

Wenn für die Datei TPMARK=YES bzw. TAPE-MARK-WRITE=*YES gilt, schreibt das DVS als Nächstes eine Abschnittsmarke. Soll keine Abschnittsmarke geschrieben werden, müssen Sie dies mit TPMARK=NO im FILE-Makroaufruf oder im FCB bzw. mit TAPE-MARK-WRITE=*BY-PROGRAM im Kommando ADD-FILE-LINK festlegen.

12.3.3 Datei ohne Kennsätze

Die Verarbeitung von Ausgabedateien ohne Kennsätze wird im Makro FILE mit dem Operanden LABEL=NO, im Kommando ADD-FILE-LINK mit dem Operanden LABEL-TYPE=*NO angefordert.

Das DVS „erwartet“, dass eine Banddatei ohne Kennsätze erstellt wird. Dennoch prüft es zunächst, ob das Band bereits Standardkennsätze enthält. Wird ein VOL1-Kennsatz gefunden, geht das DVS zur Kennsatzprüfung über, enthält das Band keinen VOL1-Kennsatz, wird es sofort positioniert, und der Benutzer kann mit der Dateiverarbeitung fortfahren.

Kennsätze prüfen

Enthält das Band einen VOL1-Kennsatz, wird zunächst die Archivnummer des Bandes überprüft (im Fehlerfall: Meldung `DMS0DA1` an den Operator). Haben Sie kein bestimmtes Band angefordert (DEVICE bzw. DEVICE-TYPE=WORK) oder ist die Datenträgerliste (VOLUME=) erschöpft, wird der Operator mit der Meldung `DMS0DFB` aufgefordert, dem Auftrag ein Band zuzuweisen.

Anschließend prüft das DVS anhand der HDR-Kennsätze der ersten Datei auf dem Band, ob Dateizugriff zulässig ist, Im Fehlerfall gibt das DVS dem Benutzer die Meldung `DM0DD7` aus, die er – falls dazu berechtigt – ignorieren kann.

Als Nächstes werden Zugriffsart (HDR3) und Freigabedatum (HDR1) überprüft. Besteht für die Datei auf dem Band noch eine Schutzfrist oder ist per „Zugriffsart“ nur Lesen erlaubt (entspricht ACCESS=READ), gibt das DVS dem Benutzer die Meldung `DMS0DA3` aus. Falls er dazu berechtigt ist, kann der Benutzer die Meldung ignorieren und mit der Dateiverarbeitung fortfahren.

Schließlich wird der Operator gefragt, ob der Benutzer auf ein Standardband eine Datei ohne Kennsätze schreiben darf. Der Operator kann dies zulassen – und so den Auftrag fortsetzen – oder eine Fehlerroutine des Benutzerprogrammes anstoßen, d.h. das Programm beenden lassen.

Positionieren und Tape Mark schreiben

Das Band wird entsprechend dem FSEQ-Wert in den Makros FILE und FCB, dem Wert FILE-SEQUENCE im Kommando ADD-FILE-LINK oder dem FSEQ-Wert im Katalogeintrag positioniert.
Gilt FSEQ=1 bzw. FILE-SEQUENCE=1, d.h. handelt es sich um die erste Datei auf dem Band, wird auf Bandanfang positioniert und – falls TPMARK=YES bzw. TAPE-MARK-WRITE=*YES gilt – eine Abschnittsmarke geschrieben.

13 CLOSE-Verarbeitung

Jede Datei, die in einem Programm eröffnet wurde, muss nach Abschluss der Dateiverarbeitung geschlossen werden. Der Programmierer kann im CLOSE-Makroaufruf entscheiden, ob er zu diesem Zeitpunkt nur eine oder mehrere im Programm noch geöffnete Dateien schließen will.

Sollte der Programmierer den CLOSE-Makroaufruf „vergessen“ haben, werden bei Programm-Beendigung automatisch alle Dateien geschlossen, die in diesem Programm geöffnet, aber noch nicht geschlossen wurden. Nur bei Systemabbruch kann es vorkommen, dass eine Datei nicht ordnungsgemäß geschlossen wird.

Der Benutzer kann solche Dateien mit dem Makro VERIF bzw. den Kommando CHECK-FILE-CONSISTENCY und REPAIR-DISK-FILES soweit wiederherstellen, dass ein Dateizugriff wieder möglich ist (siehe Abschnitt „Dateien rekonstruieren“).

13.1 CLOSE-Verarbeitung für Plattendateien

Nachdem das DVS den Abschluss evtl. noch ausstehender Schreiboperationen abgewartet hat, gibt es alle PAM-Seiten frei, die für den aufrufenden Auftrag in der zu schließenden Datei noch gesperrt sind. Als Letztes stellt es den Inhalt des Dateisteuerblocks wieder so her, wie er vor dem OPEN-Aufruf war, und gibt einen evtl. vom OPEN automatisch für die Datei angelegten TFT-Eintrag frei.

Das DVS gibt alle Arbeitsbereiche im Klasse-5-Speicher frei, die während der Dateiverarbeitung belegt waren. Dies sind z.B. die Ein-/Ausgabebereiche, die der Datei beim Eröffnen automatisch zugewiesen wurden.

Im Katalogeintrag einer zu schließenden Datei wird der Zugriffszeitpunkt (Felder ACC-DATE und ACC-TIME) und der Zugriffszähler (Feld ACC-COUNT) aktualisiert.

Für *zum Schreiben eröffnete Dateien* werden zusätzlich der Zeitpunkt der letzten Änderung (Felder CHANG-DATE und CHANG-TIME), die höchste belegte Seitennummer (Feld HIGH-US-PA) und die Nummer des höchsten gültigen Bytes im letzten logischen Block (Last Byte Pointer) aktualisiert. Die höchste belegte Seitennummer ist die letzte von der Datei belegte PAM-Seite und gibt damit die logische Dateigröße an.

Für *neu angelegte Dateien*, d.h. OPEN-Modus OUTIN bzw. OUTPUT, wird zusätzlich der Zeitpunkt der Dateierstellung (Felder CRE-DATE und CRE-TIME) und der Zeitpunkt der Freigabe (Felder EXPIR-DATE und EXPIR-TIME) eingetragen.

Die o.g. sowie weitere Felder aus dem Katalogeintrag lassen sich mit dem Kommando SHOW-FILE-ATTRIBUTES bzw. dem Makro FSTAT ermitteln.

13.2 CLOSE-Verarbeitung für Banddateien

Auch bei Banddateien läuft die CLOSE-Verarbeitung nach dem oben beschriebenen Muster ab. Allerdings müssen bei Banddateien Kennsatzschreiben und Bandpositionierung berücksichtigt werden. Das DVS schreibt für Ausgabedateien automatisch Standard-Dateiende-Kennsätze (EOF) entsprechend der LABEL-Angabe im FILE- oder FCB-Makro bzw. der Angabe LABEL-TYPE im Kommando ADD-FILE-LINK.
Anschließend wird das Band so positioniert, wie es im CLOSE-Makroaufruf gefordert wurde.

Sollen Benutzer-Kennsätze (UTL) geschrieben werden, muss das Programm die dafür erforderliche LABEND-Routine enthalten. Allerdings verzweigt das DVS nicht in die LABEND-Routine, wenn mehrere Dateien gleichzeitig geschlossen werden sollen (CLOSE ALL).

Wie bei der Dateieröffnung hängt auch beim Schließen der Dateien die Kennsatzverarbeitung von der LABEL-Angabe im FILE-Makroaufruf/FCB bzw. von der Angabe LABEL-TYPE im Kommando ADD-FILE-LINK und vom Öffnungsmodus der Datei ab.

Wird eine Datei mit der Angabe LEAVE im CLOSE-Makroaufruf geschlossen, bleibt das Band auf das logische Dateiende positioniert. Wird die folgende Datei mit demselben Dateikettungsnamen verarbeitet, wird das Band bei der OPEN-Verarbeitung auf Grund der impliziten Gerätefreigabe und dem durch FILE-Aufruf mit LINK-Angabe bzw. dem durch Angabe LINK-NAME im Kommando ADD-FILE-LINK ausgelösten Zurückspulens auf Bandanfang zurückpositioniert. Der Benutzer kann dieses Zurückpositionieren umgehen, wenn er zu Beginn der Verarbeitung einen TFT-Eintrag für eine Dummy-Datei unter einem anderen Dateikettungsnamen erzeugt oder indem er verschiedene Dateikettungsnamen für die Dateiverarbeitung verwendet.

Eingabedateien schließen

Kennsatzverarbeitung

Bei Eingabedateien (OPEN INPUT) ist keine Kennsatzbehandlung erforderlich, da keine Dateieigenschaften verändert wurden. Es wird lediglich das Band entsprechend der Angabe im CLOSE-Makroaufruf positioniert.

BYPASS-Behandlung

Eingabedateien können über den BYPASS-Operanden im FILE-Makroaufruf bzw. über den Operanden BYPASS-LABEL-CHECK im Kommando ADD-FILE-LINK positioniert werden. Es muss allerdings beachtet werden, dass die Verwendung des BYPASS-Operanden beeinflusst, wie das Band am Ende der CLOSE-Verarbeitung positioniert wird: im CLOSE-Makroaufruf muss der Operand LEAVE angegeben werden, sonst wird das Band auf den Bandanfang zurückpositioniert. Wird der Operand DISCON angegeben, wird das Band nach dem Zurückspulen entladen.

Verarbeitung von Bändern ohne EOF-/EOV-Kennsätze

Normalerweise erhält ein Benutzerauftrag, der Bänder mit Standardkennsätzen, aber ohne ordnungsgemäße EOF-/EOV-Kennsätze verarbeitet, beim Erkennen einer Abschnittsmarke einen Kennsatzfehler (EXLST-Ausgang LABERR). Da in Ausnahmefällen Bänder (fremder Hersteller) zwar Standardkennsätze, aber keine EOF-/EOV-Kennsätze enthalten, unterstützt das BS2000 die Verarbeitung solcher Bänder. Die Dateien können mit der Zugriffsmethode SAM verarbeitet werden, wenn die entsprechenden Routinen zur Fehlerbehandlung (LABERR-Fehler) vorhanden sind.

Die Verarbeitung von Banddateien mit Standardkennsätzen und ohne EOF-/EOV-Kennsätze ist nur unter folgenden Bedingungen gewährleistet:

- Die Datei muss ordnungsgemäß eröffnet werden; das bedeutet, dass zum OPEN-Zeitpunkt nicht auf eine Datei positioniert werden darf, die einer Datei ohne EOF-Kennsätze folgt.

- Die Datei darf nicht mit OPEN REVERSE oder EXTEND eröffnet werden.

Ausgabedateien schließen

Wenn eine Ausgabedatei geschlossen wird, müssen in der Regel Kennsätze geschrieben werden – entsprechend der LABEL-Angabe im FILE-Makroaufruf/FCB bzw. entsprechend der Angabe LABEL-TYPE im Kommando ADD-FILE-LINK.

Ausgabedateien mit Standardkennsätzen:
*LABEL=(STD,n) bzw. LABEL-TYPE=*STD(DIN-REVISION-NUMBER)*

Das DVS schreibt automatisch Abschnittsmarken (Tape Marks) und – außer im Fall SINOUT eröffneter Dateien bei BTAM – die Kennsätze EOF1 bis EOF3; die Kennsätze EOF4 bis EOF9 werden im BS2000 nicht geschrieben. Bei BTAM-Dateien, die im Modus SINOUT eröffnet wurden, können keine Ende-Kennsätze geschrieben werden, weil zum OPEN-Zeitpunkt keine Kennsatzbehandlung erfolgte und daher keine Informationen zum Erstellen von Ende-Kennsätzen vorliegen.
Soll die Datei Benutzer-Dateiende-Kennsätze (UTL) erhalten, muss das Programm sie in einer LABEND-Routine erstellen. Da die BLIM-Funktion bzw. die BLOCK-LIMIT-Funktion im Kommando ADD-FILE-LINK nicht zum Tragen kommt, wird keine Fixpunktdatei erzeugt. Das Band wird entsprechend den Angaben im CLOSE-Makroaufruf positioniert.
Gilt DESTOC=YES aus dem FILE-Makro bzw. DESTROY-OLD-CONTENTS=*YES aus dem Kommando ADD-FILE-LINK oder DESTROY=YES aus dem Katalogeintrag, wird der weitere Bandinhalt gelöscht.

Ausgabedateien mit Nicht-Standardkennsätzen:
*LABEL=NSTD bzw. LABEL-TYPE=*NON-STD*

Nicht-Standardkennsätze müssen in einer LABEND-Routine des Benutzerprogramms erzeugt werden. Bevor es die Kennsatzroutine aktiviert, schreibt das DVS eine Abschnittsmarke (Tape Mark). Gilt TPMARK=YES aus dem Makro FILE bzw. TAPE-MARK-WRITE= *YES aus dem Kommando ADD-FILE-LINK, schreibt das DVS eine Doppel-Abschnittsmarke. Gilt TPMARK=NO im TFT-Eintrag und im Dateisteuerblock, schreibt das DVS nur eine Abschnittsmarke. Danach wird das Band entsprechend den Angaben im CLOSE-Makroaufruf positioniert. Gilt DESTOC=YES aus Makro FILE bzw. DESTROY-OLD-CONTENTS= *YES aus dem Kommando ADD-FILE-LINK oder DESTROY=YES aus dem Katalogeintrag, wird der weitere Bandinhalt gelöscht.

Ausgabedateien ohne Kennsätze:
*LABEL=NO bzw. LABEL-TYPE=*NO*

Das DVS schreibt die Doppel-Abschnittsmarke für Datei- und Bandende.
Gilt aus dem Makro FILE DESTOC=YES oder aus dem Kommando ADD-FILE-LINK DESTROY-OLD-CONTENTS=*YES oder aus dem Katalogeintrag DESTROY=YES, wird der weitere Bandinhalt gelöscht.

14 EOV-Verarbeitung

Die EOV-Verarbeitung (End-of-Volume) wird für SAM-Banddateien angestoßen, wenn beim Lesen oder Schreiben einer Banddatei, die sich über mehrere Bänder erstreckt, ein Bandwechsel durchgeführt werden muss. Sie kann sowohl für BTAM- als auch für SAM-Banddateien explizit mit dem FEOV-Makroaufruf angestoßen werden. Wie die Kennsatzverarbeitung beim Öffnen oder Schließen einer Banddatei hängt auch die EOV-Verarbeitung vom Eröffnungsmodus der Datei (Ein- oder Ausgabedatei) und von der LABEL-Angabe in FILE/FCB bzw. von der Angabe LABEL-TYPE im Kommando ADD-FILE-LINK ab.

Der Aufbau der Routinen, in denen UTL-, UVL-, UHL-Kennsätze erzeugt werden, ist im Abschnitt „EXLST-Ausgänge der Kennsatz-Verarbeitung für Banddateien“ beschrieben.

14.1 Eingabedateien

Die EOV-Verarbeitung wird implizit angestoßen, wenn das DVS statt eines Datenblocks/ Datensatzes eine Abschnittsmarke erkennt. Der Verlauf der EOV-Verarbeitung hängt u.a. vom BYPASS-Operanden des FILE-Makroaufrufs bzw. dem Operanden BYPASS-LABEL-CHECK im Kommando ADD-FILE-LINK ab (siehe Abschnitt „BYPASS-Behandlung“). Wird sie explizit angestoßen, positioniert das DVS das Band hinter die nächste Abschnittsmarke (Tape Mark).

Welche Kennsätze bei der EOV-Verarbeitung, d.h. beim Bandwechsel, geprüft werden, hängt von der Leserichtung ab. Wird die Datei „normal“ vom Dateianfang in Richtung Dateiende gelesen, werden EOF-/EOV-/UTL-Kennsätze geprüft. Wird die Datei rückwärts gelesen (REVERSE), werden die HDR- und UHL-Kennsätze wie Band-/Dateiende-Kennsätze behandelt. Treten Fehler auf, aktiviert das DVS eine CLOSE-Fehlerroutine.

14.1.1 Datei mit Standardkennsätzen

Die Verarbeitung von Standardkennsätzen wird angefordert:

- im Makro FILE mit LABEL=(STD,n)
- im Kommando ADD-FILE-LINK mit LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n)

Das DVS prüft den auf die Abschnittsmarke folgenden Satz:

- Ist dieser weder ein Dateiende- noch Bandende-Kennsatz (EOF1 oder EOV1), aktiviert das DVS die LABERR-Routine. Ist der LABERR-Ausgang des EXLST-Makroaufrufs nicht versorgt, wird dem Benutzer eine entsprechende Fehlermeldung ausgegeben (DMSODE9).
  Der Benutzer kann sein Programm dann per CLOSER-Routine beenden oder die Dateiverarbeitung trotz des aufgetretenen Fehlers fortsetzen.
- Folgen auf die Abschnittsmarke Dateiende- oder Bandende-Kennsätze sowie eine Doppel-Abschnittsmarke, vergleicht das DVS den Inhalt des Kennsatzfeldes „Blockzähler“ (in EOF1 oder EOV1), mit dem aktuellen DVS-internen Blockzähler. Stimmen diese Zahlen nicht überein, aktiviert das DVS eine LABERR-Routine im Benutzerprogramm oder gibt eine Fehlermeldung aus (`DMS0DE6`).
  Der Benutzer kann sein Programm per CLOSER-Routine beenden oder die Verarbeitung fortsetzen.

Weitere Standard-Datei- oder Bandende-Kennsätze werden übergangen. Enthält das Band Benutzer-Ende-Kennsätze (UTL), aktiviert das DVS die LABEND- oder LABEOV-Routine im Benutzerprogramm. Es unterscheidet Dateiende- und Bandende-Kennsätze anhand der vorangegangenen Standardkennsatzgruppe (EOF oder EOV).

Hat das DVS Dateiende-Kennsätze (EOF) gelesen, veranlasst es nur dann einen Bandwechsel, wenn im Makro FILE mit dem Operanden TVSN bzw. wenn im Kommando ADD-FILE-LINK mit dem Operanden PROCESS-VOLUME eine temporäre Datenträgerliste angelegt wurde und diese Liste noch weitere Archivnummern enthält. Ist diese Liste erschöpft oder wurde im Makro- oder Kommandoaufruf der entsprechende Operand gar nicht angegeben, erkennt das DVS „Dateiende“. Das Band ist zu diesem Zeitpunkt auf die Abschnittsmarke hinter den Dateiende-Kennsätzen positioniert.

Hat das DVS Bandende-Kennsätze (EOV) gelesen, veranlasst es einen Bandwechsel, wenn die Datenträgerliste der TFT noch weitere Archivnummern enthält. Die TFT-Datenträgerliste wird durch den VSEQ-Operanden des FILE-Makroaufrufs bzw. durch den Operanden VOL-SEQUENCE-NUMBER des Kommandos ADD-FILE-LINK beeinflusst. Ist die Eingabedatei eine FOREIGN-Datei, wird auf jeden Fall Bandwechsel veranlasst.

Das DVS aktiviert eine EOVCTRL-Routine (falls im EXLST-Makroaufruf definiert), in der der Benutzer z.B. Fixpunktschreiben veranlassen kann; die EOVCTRL-Routine wird mit einem EXRTN-Makroaufruf beendet.

Der Bandwechsel muss vom Operator durchgeführt werden. Das DVS gibt dazu an der Konsole eine Meldung aus (DMS0DE4). Der Operator kann jetzt das Nachfolgeband zuweisen und die Verarbeitung fortsetzen oder das Benutzerprogramm abbrechen (bzw. den NODEV-Ausgang des EXLST-Makroaufrufs aktivieren).

Nach Abschluss des Bandwechsels wird für das Folgeband die Bandanfangs-Behandlung angestoßen (siehe Abschnitt „Kennsatz-Verarbeitung für Eingabedateien“).

14.1.2 Datei mit Nicht-Standardkennsätzen

Die Verarbeitung von Nicht-Standardkennsätzen wird angefordert:

- im Makro FILE mit LABEL=NSTD
- im Kommando ADD-FILE-LINK mit LABEL-TYPE= *NON-STD

Das DVS steuert die EOV-Verarbeitung durch Aktivieren der entsprechenden Routinen des Benutzerprogramms, die die Kennsatzverarbeitung übernehmen.

Hat das DVS die Abschnittsmarke erkannt, aktiviert es die LABEOV-Routine, in der die Bandende-Kennsätze ausgewertet werden. Nach Auswertung der Kennsätze führt – falls erforderlich – das DVS den Bandwechsel durch mit Meldung an den Operator usw. Für das neu zugewiesene Band wird dann die Bandanfangs-Behandlung durchgeführt, bevor das Benutzerprogramm mit dem Lesen der Datei fortfahren kann.

War kein Bandwechsel erforderlich, aktiviert das DVS die LABEND-Routine zur Auswertung von Dateiende-Kennsätzen. Anschließend positioniert das DVS das Band um eine Abschnittsmarke zurück und leitet die Dateiende-Behandlung ein (EOFADDR-Ausgang des EXLST-Makroaufrufs).

14.1.3 Datei ohne Kennsätze

Die Verarbeitung ohne Kennsätze wird angefordert:

- im Makro FILE mit LABEL=NO
- im Kommando ADD-FILE-LINK mit LABEL-TYPE=*NO

Das DVS prüft, ob eine weitere Abschnittsmarke folgt und ob in der Datenträgerliste aus dem FILE-Makroaufruf bzw. aus dem Kommando ADD-FILE-LINK noch weitere Archivnummern enthalten sind.

Enthält das Band keine Doppel-Abschnittsmarke und ist die Datenträgerliste noch nicht erschöpft, veranlasst das DVS den Bandwechsel. Anschließend führt es die Bandanfangs-Behandlung durch, bevor das Programm die Datei weiter verarbeiten kann.

Hat das DVS auf dem Band eine Doppel-Abschnittsmarke gefunden oder wird kein weiteres Band benötigt, positioniert es das Band hinter die zuletzt gelesene Abschnittsmarke und aktiviert eine Dateiende-Routine im Benutzerprogramm (EOFADDR-Ausgang des EXLST-Makroaufrufs).

14.1.4 BYPASS-Behandlung

Beim Erkennen einer Abschnittsmarke in einer SAM-Datei wird von SAM die EOV-Verarbeitung systemintern angestoßen; sie kann auch durch einen FEOV-Makroaufruf angestoßen werden.

Die Dateiende-Kennsatz-Prüfung wird unterdrückt, sodass nicht zwischen EOF- und EOV-Kennsätzen unterschieden wird und bei Dateien ohne Standardkennsätze die Doppel-Abschnittsmarke, die der letzten Datei auf dem Band folgt, nicht erkannt wird.

Ein Bandwechsel wird nur dann durchgeführt, wenn die TFT-Datenträgerliste noch eine weitere Archivnummer enthält. In diesem Fall wird die Bandposition in Abhängigkeit von der TPMARK- bzw. TAPE-MARK-WRITE-Angabe auf den Datenanfang des neuen Datenträgers gesetzt :

- TPMARK=NO an den Bandanfang
- TPMARK=YES hinter die erste Abschnittsmarke

Bei LABEL=(STD,n) bzw. LABEL-TYPE=*STD(DIN-REVISION-NUMBER) wird TPMARK nicht ausgewertet.

Enthält die TFT-Datenträgerliste keine weitere Archivnummer (Dateiende), wird das Band hinter die gelesene Abschnittsmarke positioniert.

14.2 Ausgabedateien

- Datei mit Standardkennsätzen
- Datei mit Nicht-Standardkennsätzen
- Datei ohne Kennsätze

14.2.1 Datei mit Standardkennsätzen

Die Verarbeitung von Standardkennsätzen wird angefordert:

- im Makro FILE mit LABEL=(STD,n)
- im Kommando ADD-FILE-LINK mit LABEL-TYPE=*STD(DIN-REVISION-NUMBER=n)

Die EOV-Verarbeitungsroutine wird bei SAM implizit angestoßen, wenn das DVS eine Bandendemarke erkennt oder das im FILE-/FCB-Aufruf mit dem Operanden BLIM bzw. im Kommando ADD-FILE-LINK mit dem Operanden BLOCK-LIMIT angegebene Blocklimit erreicht ist. Sie kann auch explizit mit FEOV-Makroaufruf angestoßen werden.

Verarbeitungsschritte

- ausstehende Ein-/Ausgaben bearbeiten; bei SAM-Dateiverarbeitung ist der letzte Block, der auf das Band geschrieben wird, eventuell nur teilweise gefüllt
- Abschnittsmarke schreiben
- EOV-Kennsätze schreiben (entsprechend der HDR-Kennsätze)
- UTL-Kennsätze schreiben (LABEOV-Routine)
- Doppel-Abschnittsmarke schreiben
- falls gefordert: Fixpunkt an das Bandende schreiben
- Wenn DESTOC=YES (aus FILE-Aufruf) bzw. DESTROY-OLD-CONTENTS (aus Kommando ADD-FILE-LINK) oder DESTROY=YES (im Katalogeintrag) gilt: Restdaten auf dem Band überschreiben
- Bandwechsel einleiten; entsprechend der Datenträgerliste in TFT/TST wird das Folgeband oder beim Operator ein freies Band angefordert
- Bandanfangs-Behandlung für das Folgeband einleiten

Wenn eine Nicht-BS2000-Banddatei erweitert wird (OPEN INOUT/EXTEND), ist zu beachten, dass der bereits bestehende Teil der Datei Standardkennsätze enthalten kann, die vom BS2000 nicht unterstützt werden, z.B. HDR4-HDR9. Das BS2000 schreibt maximal drei EOV-Kennsätze (EOV1-EOV3) an das Ende jedes neuen Dateiabschnitts und maximal drei HDR-Kennsätze (HDR1-HDR3) an den Beginn der folgenden Dateiabschnitte (entsprechend maximal drei EOF-Kennsätze am Dateiende).

14.2.2 Datei mit Nicht-Standardkennsätzen

Die Verarbeitung von Nicht-Standardkennsätzen wird angefordert:

- im Makro FILE mit LABEL=NSTD
- im Kommando ADD-FILE-LINK mit LABEL-TYPE=*NON-STD

Die EOV-Verarbeitungsroutine wird implizit beim Erkennen der Bandendemarke angestoßen oder explizit mit FEOV-Makroaufruf.

Verarbeitungsschritte

- ausstehende Ein-/Ausgaben bearbeiten und den letzten Block auf das Band schreiben
- Abschnittsmarke schreiben, wenn TPMARK=YES bzw. TAPE-MARK-WRITE=*YES gilt
- Verzweigen zur LABEOV-Routine im Benutzerprogramm
- Doppel-Abschnittsmarke schreiben, wenn TPMARK/TAPE-MARK-WRITE=*YES gilt
- falls DESTOC=YES im Makro FILE bzw. DESTROY-OLD-CONTENTS im Kommando ADD-FILE-LINK angegeben wurde oder aus dem Katalogeintrag DESTROY=YES gilt: den Rest des Bandes löschen
- Bandwechsel durchführen: ein Folgeband entsprechend den Angaben in der Datenträgerliste oder ein freies Band beim Operator anfordern
- Bandanfangs-Behandlung für das Folgeband durchführen

Erst nach der Bandanfangs-Behandlung für das neu zugewiesene Band wird die Dateiverarbeitung fortgesetzt.

14.2.3 Datei ohne Kennsätze

Die Verarbeitung ohne Kennsätze wird angefordert:

- im Makro FILE mit LABEL=NO
- im Kommando ADD-FILE-LINK mit LABEL-TYPE=*NO

Die EOV-Verarbeitungsroutine wird implizit angestoßen, wenn das DVS die Bandendemarke erkennt, oder explizit mit FEOV-Makroaufruf.

Verarbeitungsschritte

- ausstehende Ein-/Ausgaben bearbeiten und den letzten Block auf das Band schreiben
- Doppel-Abschnittsmarke schreiben
- falls im FILE-Aufruf DESTOC=YES bzw. DESTROY-OLD-CONTENTS im Kommando ADD-FILE-LINK angegeben wurde oder aus dem Katalogeintrag DESTROY=YES gilt: den Rest des Bandes löschen
- Bandwechsel einleiten: Folgeband entsprechend der Datenträgerliste oder freies Band beim Operator anfordern
- Bandanfangs-Behandlung durchführen

Erst nach Abschluss der Bandanfangs-Behandlung für das Folgeband wird die Dateiverarbeitung fortgesetzt.

15 Behandlung von Fehlern und Sonderereignissen

Bei der Bearbeitung einer Datei können sich Situationen ergeben, die eine Entscheidung verlangen, z.B. wenn beim Lesen das Dateiende erreicht wird, wenn ein Fehler auftritt oder sich bei Simultan-Aktualisierung Konflikte mit anderen Aufträgen ergeben.

Damit sich das DVS in einer solchen Situation mit dem Benutzerauftrag verständigen kann, ist im FCB-Makroaufruf eine Exit-Adresse anzugeben (Operand EXIT im FCB-Makroaufruf). Ist diese Adresse nicht vorhanden, bricht das DVS den Programmlauf ab.

15.1 Exit-Adresse im FCB

Verweis auf globale Programm-Routine

Die globale Routine ist für die Bearbeitung aller überhaupt vorkommenden Fehler zuständig. Die Adresse ist in diesem Fall in Klammern einzuschließen, also EXIT=(adr).

Die Exit-Routine kann die FCB-Felder ID1XITB und ID1ECB auswerten: ID1XITB zeigt die Fehlerursache an, ID1ECB enthält einen DVS-Fehlerschlüssel (siehe auch Abschnitt „Fehlermeldungsschlüssel“ im Anhang des Handbuches „DVS-Makros“ [1]).

Die Routine kann auch genutzt werden, das Programm ordnungsgemäß zu beenden (Dateien schließen!) oder Speicherabzüge anzufordern (Makros `CDUMP2` oder `TERM, DUMP=Y`, siehe Handbuch „Makroaufrufe an den Ablaufteil“ [2]).

Verweis auf den EXLST-Makroaufruf

Mit der Exit-Adresse im FCB kann auch ein EXLST-Makroaufruf adressiert werden.
In diesem Fall ist im EXIT-Operanden die Adresse ohne Klammern anzugeben: EXIT=adr. Der EXLST-Makroaufruf kann genutzt werden, um einerseits spezifische Ereignisse/Fehler abzufragen und Sondermaßnahmen einzuleiten, während andere Ereignisse/Fehler global in einer gemeinsamen Routine behandelt werden.

Ereignisse, für die kein separater EXLST-Ausgang definiert werden soll, können über den COMMON-Operanden „abgefangen werden“. Die COMMON-Routine führt dann Fehlerbehandlung usw. durch. Gleichzeitig wird durch Definition des COMMON-Operanden auch Programmabbruch vermieden, wenn ein Ereignis eintritt, für das kein separater EXLST-Ausgang versorgt wurde.

> **Hinweis**
>
> Wird zum OPEN-Zeitpunkt bei der FCB-Validierung erkannt, dass der FCB nicht korrekt ist (z.B. Speicherbereich nicht allokiert, mit derselben FCB-Adresse ist bereits eine Dateibearbeitung aktiv, usw.) kommt dieser Mechanismus nicht zum tragen. In einem solchen Fall erfolgt immer eine abnormale Programmbeendigung mit dem Fehlerschlüssel `DMS0D9F`.

15.2 Exit-Routinen: EXLST-Makroaufruf

Mit den Operanden des EXLST-Makroaufrufs können Sie für jedes Ereignis einen Fehlerausgang definieren, d.h. die Adresse eines Programmteils angeben, in dem eine Fehlerbehandlung durchgeführt oder zumindest das Programm ordnungsgemäß beendet wird (Schließen geöffneter Dateien ...). Bei Verwendung von EXLST-Ausgängen kann die Auswertung des FCB-Feldes ID1XITB entfallen, da jeder EXLST-Ausgang genau einem Ereignis entspricht.

Das DVS überträgt die Adresse des Dateisteuerblocks in das Mehrzweckregister 1, bevor es die Steuerung an das Benutzerprogramm abgibt. Dadurch ist es möglich, für verschiedene Dateien dieselben Programmteile für die Fehlerbehandlung zu benutzen.

Das DVS überträgt die Adresse des Befehls, der bei normaler Verarbeitung als nächster ausgeführt worden wäre, in das Feld ID1RTNAD des Dateisteuerblocks. Diese Adresse verweist auf den nächsten Befehl des Benutzerprogramms.

Die Art des Ereignisses steht codiert im Feld ID1XITB des Dateisteuerblocks. Der jeweilige Code ist der Tabelle zu entnehmen (siehe Handbuch „DVS-Makros“ [1]).

Bei den EXLST-Ausgängen EOVCTRL, ERROPT, LABERR, OPENV, OPENX, OPENZ und WLRERR ist zur Fortsetzung des Programms der EXRTN-Makroaufruf anzugeben, bei LABEND, LABEOV, LABGN der LBRET-Makroaufruf.

Bei Erreichen eines Sonderzustands während der Bearbeitung einer Datei versorgt das DVS die Felder ID1XITB und ID1ECB im zugehörigen TU-FCB. Sie werden nicht automatisch zurückgesetzt, sondern nur bei erneutem Auftreten eines Sonderzustands aktualisiert.

15.2.1 EXLST-Ausgänge der OPEN-Verarbeitung

Während der OPEN-Verarbeitung können zum einen beim Eröffnen der Datei Fehler auftreten, die über EXLST-Operanden abgefangen werden können. Zum anderen können Sie an bestimmten Stellen der OPEN-Verarbeitung den FCB verändern oder Bandkennsätze erzeugen – ebenfalls gesteuert über EXLST-Operanden.

LOCK	die Datei soll schreibend eröffnet werden, sie ist jedoch bereits von einem anderen Benutzer eröffnet worden.
NODEV	der Datenträger, auf dem die Datei liegt, kann nicht bereitgestellt werden.
NOSPACE	für die Sekundärzuweisung bei OPEN=EXTEND ist unzureichender Speicherplatz vorhanden.
OPENC	die Datei war als Ausgabedatei eröffnet und wurde nicht ordnungsgemäß geschlossen.
OPENER	Fehler beim Eröffnen der Datei; der Benutzer kann anhand des Fehlerschlüssels im FCB den Fehler analysieren.
OPENV	für Banddateien: im Benutzerprogramm werden UVL-Kennsätze bzw. Nicht-Standard-Bandanfangs-Kennsätze überprüft oder geschrieben.
OPENX	der FCB-Inhalt kann überprüft und geändert werden; die Änderungen gehen (bei Ausgabedateien) in Katalogeintrag und Kennsätze ein.
OPENZ	für Ausgabedateien kann der FCB verändert werden, ohne dass die Änderungen in Katalogeintrag oder Kennsätze eingehen.
PASSER	für eine geschützte Datei wurde das falsche Kennwort angegeben.

15.2.2 EXLST-Ausgänge der Kennsatz-Verarbeitung für Banddateien

Für alle hier beschriebenen Ausgänge gilt Folgendes:

- Das DVS stellt dem Benutzer im Register 0 die Adresse eines 80 Byte großen Pufferbereichs zur Verfügung und überträgt bei Eingabedateien den Inhalt der Benutzer-Kennsätze an diese Adresse. Bei Ausgabedateien übernimmt es den Inhalt der Benutzer-Kennsätze von dieser Adresse.
- Zur Fortsetzung des Programms muss bei der Behandlung von UVL-, UHL- und UTL-Kennsätzen der LBRET-Makro aufgerufen werden, sonst der EXRTN-Makro.
- Bei Dateien mit Nicht-Standardkennsätzen werden die Band- und Datei-Kennsätze mit BTAM-Makroaufrufen bearbeitet, unabhängig von der gewählten Zugriffsmethode.
- Bei Standard-Benutzer-Kennsätzen erfolgt das Schreiben automatisch.

EXLST-Ausgänge steuern das Schreiben von Benutzer- bzw. Nicht-Standardkennsätzen, auch Fehler, die bei Kennsatzprüfungen auftreten, können abgefangen werden.

OPENV	im Benutzerprogramm werden UVL-Kennsätze bzw. Nicht-Standard-Bandanfangs-Kennsätze überprüft oder geschrieben.
LABGN	im Benutzerprogramm können UHL-Kennsätze bzw. Nicht-Standard-Dateianfangs-Kennsätze überprüft oder geschrieben werden.
LABEND	Im Benutzerprogramm können Benutzer-/Nicht-Standard-Dateiende-Kennsätze überprüft bzw. geschrieben werden. Das DVS unterscheidet UTL-Datei- und UTL-Bandende-Kennsätze anhand der vorangegangenen System-Kennsatzgruppe (EOF, EOV).
LABEOV	Im Benutzerprogramm können Benutzer-Bandende-Kennsätze oder Nicht-Standard-Bandende-Kennsätze überprüft/geschrieben werden. Die EOV-Verarbeitung wird entweder automatisch angestoßen (nur bei SAM) oder über einen FEOV-Makroaufruf (SAM und BTAM). Bei BTAM-Verarbeitung muss der Benutzer das Feld ERRBYTE auswerten, in dem „Bandende“ angezeigt wird, und dann mit dem FEOV-Makroaufruf Bandwechsel einleiten – wenn er nicht schon vorzeitig Bandwechsel veranlasst hat.
LABERR	Die LABERR-Routine wird aktiviert, wenn bei der Verarbeitung von Dateien mit Standardkennsätzen ein Kennsatzfehler auftritt, z.B. wenn EOF-/EOV-Kennsätze fehlen oder für Eingabedateien der Blockzähler im EOF-/EOV-Kennsatz nicht mit dem DVS-internen Blockzähler übereinstimmt. Der Benutzer muss Register 0 mit einem „Aktionsschlüssel“ versorgen, der die weitere Verarbeitung steuert. Die LABERR-Routine wird mit dem EXRTN-Makroaufruf beendet. Ist der LABERR-Ausgang nicht versorgt, gibt das DVS eine Fehlermeldung aus. Der Benutzer kann dann entscheiden, ob der Programmlauf fortgesetzt oder ob eine CLOSE-Fehlerroutine aktiviert werden soll.

15.2.3 EXLST-Ausgänge und Aktionsmakro-Aufrufe

Die folgende Tabelle gibt einen Überblick über mögliche Aktionsmakros, die Sie in den verschiedenen EXLST-Ausgängen aufrufen können.

Nähere Informationen zu den Fehlerausgängen und den Aktionsmakros finden Sie im Handbuch „DVS-Makros“ [1].

Fehlerausgang	Mögliche Aktionsmakros
CLOSEPOS	BTAM, CLOSE, EXRTN
DLOCK	FCB
DUPEKY	INSRT, PUT
EOFADDR	CLOSE, GET, GETR, GETFL (LIMIT = END), FEOV
ERRADDR	BTAM, FILE
ISPERR	FILE
LABEND	LBRET, CLOSE, EXRTN
LABEOV	FEOV, EXRTN, LBRET
LABERR	EXTRN, CLOSE
LABGN	LBRET, OPEN, EXRTN
LOCK	OPEN
NOFIND	GETKY, ELIM (mit KEY-Angabe), GETFL (LIMIT=KEY)
NOSPACE	OPEN (OUTPUT/EXTEND)
OPENC	VERIFY
OPENV	EXRTN, BTAM
PGLOCK	PUTX, ELIM (ohne KEY-Angabe), RETRY, FCB
SEQCHK	PUT
USERERR	PUTX, ELIM (ohne KEY-Angabe), GETFL, SETL

Tabelle 31: EXLST-Ausgänge und Aktionsmakro-Aufrufe

15.3 DMS-Fehlerschlüssel

Die DMS-Fehlerschlüssel ermöglichen auch auf Programmebene eine genaue Analyse des aufgetretenen Fehlers; sie werden im FCB-Feld ID1ECB hinterlegt.

Eine Dokumentation der DVS-Fehlerschlüssel ist im IDEMS-Makroaufruf enthalten.
Die Texte, die den jeweiligen Fehler beschreiben, sind dort allerdings sehr knapp. Ausführlichere Informationen können anhand des DVS-Fehlerschlüssels mit dem Kommando bzw. der SDF-Standardanweisung HELP-MSG-INFORMATION angefordert werden (siehe Handbuch „Kommandos“ [3]).

16 Zugriffsmethoden

Die Zugriffsmethoden des DVS übertragen Daten zwischen Datei und Adressraum des Auftrags. Technisch ist dies eine Datenübertragung zwischen Peripheriespeicher und Hauptspeicher der Zentraleinheit. Dabei übernimmt das DVS die Behandlung der Geräte und Datenträger und bietet dem Benutzer Schnittstellen für den Zugriff auf Datensätze bzw. auf Datenblöcke.

Beim Zugriff eines Benutzerprogramms auf den Inhalt einer Datei lassen sich satz- und blockorientierter Zugriff unterscheiden sowie die Betriebsarten Übertragungsbetrieb und Ortungsbetrieb.

Bei allen Zugriffsmethoden überträgt das DVS Datenblöcke zwischen Peripherie und Hauptspeicher. Der Teil des Hauptspeichers, der einen Datenblock aufnimmt oder dessen Inhalt zum Peripheriespeicher übertragen wird, heißt Puffer. Der Puffer gehört zum Adressraum des Auftrags, der die Ein-/Ausgabeoperation veranlasst hat. Bei der Verarbeitung mit NK-ISAM liegen die Puffer in ISAM-Pools (siehe "ISAM-Pools"). Wird eine Datei satzweise im Übertragungsbetrieb verarbeitet, überträgt das DVS die Datensätze vom Puffer in den Arbeitsbereich des Programms.

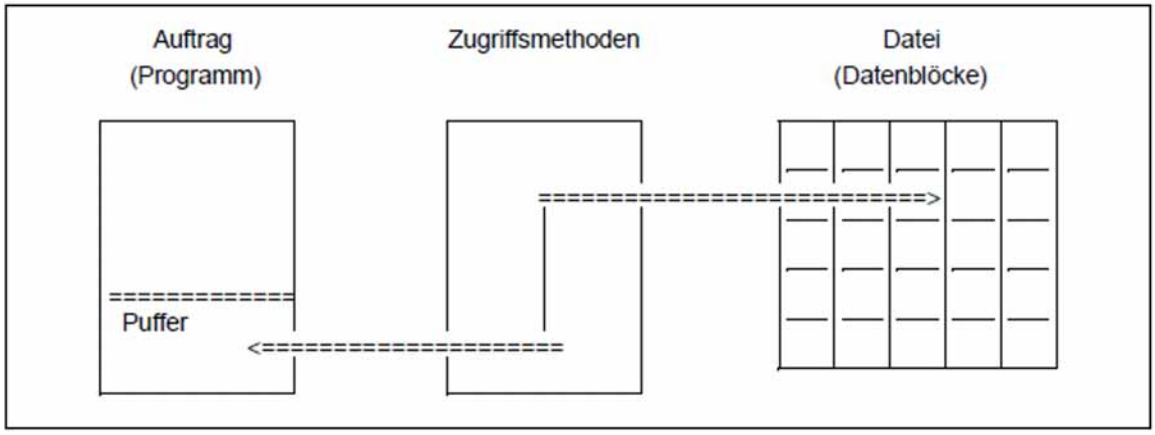


Bild 19: Übertragung von Datenblöcken

16.1 Privilegierte Zugriffsmethoden (PPAM/PTAM)

Die zentralen Zugriffsmethoden des BS2000 sind die privilegierte Plattenzugriffsmethode PPAM (Privileged Primary Access Method) und die privilegierte Bandzugriffsmethode PTAM (Privileged Tape Access Method).
PPAM und PTAM werden vom DVS zur Ausführung von Dateizugriffen genutzt. Das DVS gibt Dateizugriffsanforderungen an PPAM/PTAM weiter.

Alle Zugriffe auf Plattendateien werden von PPAM durchgeführt. PPAM arbeitet mit Standardblöcken, den sog. PAM-Seiten (auch PAM-Block, im Englischen PAM-Page oder auch HP=Half Page). Es ist den Zugriffsmethoden ISAM, SAM, UPAM, FASTPAM, DIV und EAM bei Plattendateien vorgeschaltet.

Alle Zugriffe auf Banddateien werden von PTAM durchgeführt. PTAM unterstützt die Bandzugriffsmethode BTAM sowie die Zugriffsmethoden SAM und UPAM, wenn mit SAM und UPAM Banddateien verarbeitet werden. PTAM arbeitet unabhängig von PPAM. Aus Sicht von PTAM besteht der Inhalt eines Magnetbandes aus einer Folge von Datenblöcken und Band-Abschnittsmarken.

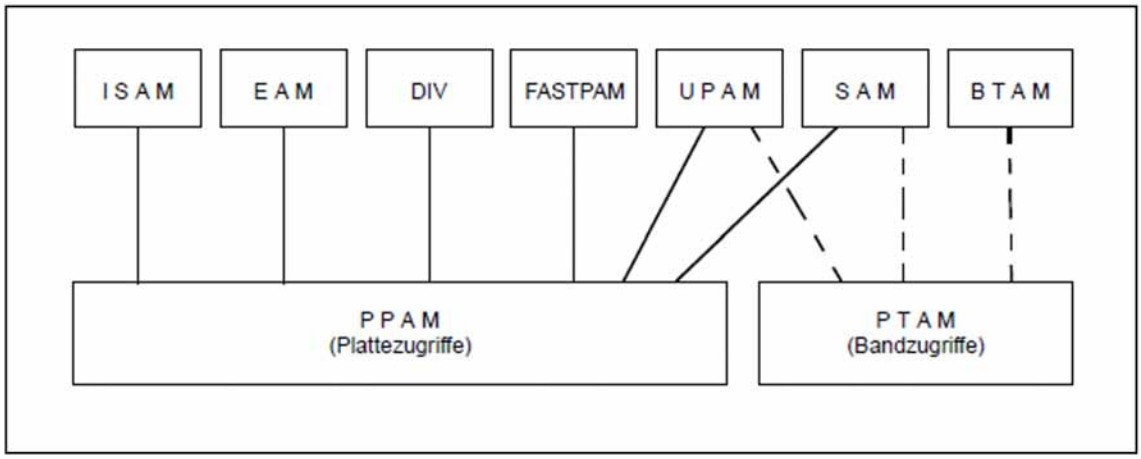


Bild 20: PPAM/PTAM

16.2 Übersicht über die nicht-privilegierten Zugriffsmethoden

Die Wahl einer Zugriffsmethode hängt ab von den Anforderungen, die sich aus dem jeweils zu lösenden Problem ergeben und den speziellen Möglichkeiten der einzelnen Zugriffsmethoden. Die einzelnen Zugriffsmethoden sind in den entsprechenden Kapiteln ausführlich beschrieben. Im Folgenden wird ein Überblick über die Eigenschaften der nicht-privilegierten Zugriffsmethoden gegeben.

Auswahlkriterium Datenträger

Datenträger	ISAM	SAM	BTAM	UPAM	FASTPAM	DIV	EAM
gemeinschaftliche Platte							
• K-Platte	x	x	-	x	x	x	x
• NK2-Platte	x	x	-	x	x	x	x
• NK4-Platte	x	x	-	x	x	x	
Privatplatte							
• NK2-Platte	x	x	-	x	x	x	x
• NK4-Platte	x	x	-	x	x	x	x
Band		x	x	x			

Tabelle 32: Kriterium Datenträger bei Wahl der Zugriffsmethode

Auswahlkriterium Zugriffsart

Zugriffsart	ISAM	SAM	BTAM	UPAM	FASTPAM	DIV	EAM
blockorientiert	-	-	x	x	x	x	x
satzorientiert	x	x	-	-	-	-	-

Tabelle 33: Kriterium Zugriffsart bei Wahl der Zugriffsmethode

Auswahlkriterium Satztyp

Satztyp	ISAM	SAM
variable	x	x
fixed	x	x
undefined	-	x

Tabelle 34: Kriterium Satztyp bei Wahl der Zugriffsmethode

Auswahlkriterium Festlegung der Datenstruktur

Festlegung der Datenstruktur	ISAM	SAM	BTAM	UPAM	FASTPAM	DIV	EAM
durch DVS	x	x	-	-	-	-	-
durch Benutzer	-	-	x	x	x	x	-

Tabelle 35: Kriterium Datenstruktur bei Wahl der Zugriffsmethode

Auswahlkriterium Lebensdauer der Datei

Lebensdauer der Datei	ISAM	SAM	BTAM	UPAM	FASTPAM	DIV	EAM
permanente Datei	x	x	x	x	x	x	
temporäre Datei	-	-	-	-	-	-	x

Tabelle 36: Kriterium Datei-Lebensdauer bei Wahl der Zugriffsmethode

Auswahlkriterium Multi-User-Betrieb

Multi-User-Betrieb	ISAM	SAM	BTAM	UPAM	FASTPAM	DIV	EAM
Verarbeitung	x	-	-	x	x	x	-

Tabelle 37: Kriterium Multi-User-Betrieb bei Wahl der Zugriffsmethode

Vorteile und Einsatzmöglichkeiten der einzelnen Zugriffsmethoden

ISAM – Indexed Sequential Access Method

ISAM ist eine satzorientierte Zugriffsmethode für Plattendateien, mit der auf Sätze über vorher definierte Schlüssel (Primär-, Sekundärschlüssel) zugegriffen werden kann. Mit ISAM können Dateien auch sequenziell verarbeitet werden.

SAM – Sequential Access Method

Die Zugriffsmethode SAM unterstützt das sequenzielle Lesen und Schreiben von logischen Datensätzen.

UPAM – User Primary Access Method

UPAM ist eine blockorientierte Zugriffsmethode im BS2000 für wahlfreien Zugriff auf Plattendateien. Zu jedem Zeitpunkt kann auf einen beliebigen Block der Datei lesend oder schreibend zugegriffen werden.
Grundlage der Verarbeitung ist der Datenblock; Satzstrukturen werden nicht erkannt.

BTAM – Basic Tape Access Method

Die Zugriffsmethode BTAM dient zum Abspeichern und Wiederauffinden von blockorientierten Daten in einer sequenziell organisierten Banddatei. Über BTAM können auch Banddateien verarbeitet werden, die nicht in einem BS2000-System erstellt wurden, sofern sie den allgemein gültigen Hardwarekonventionen für Datenaufzeichnung bei Magnetbändern entsprechen.

FASTPAM – Fast Primary Access Method

FASTPAM ist eine Blockzugriffsmethode für NK4-Plattendateien. Es zeichnet sich aus durch:

- geringe Pfadlänge bei der Durchführung von Ein-/Ausgaben
- eine schmale, übersichtliche Schnittstelle

- Unterstützung von Ein-/Ausgaben in Datenräumen.

DIV – Data-In-Virtual

DIV ist eine Zugriffsmethode, die sich von den Zugriffsmethoden ISAM, SAM oder UPAM dadurch unterscheidet, dass sie ohne eine Strukturierung von Dateien in Sätze oder Blöcke, ohne Ein- bzw. Ausgabepuffer und ohne Operationen wie GET, PUT usw. auskommt.

Bereiche einer Datei oder eine ganze Datei werden in einen virtuellen Adressraum mithilfe der DIV-Funktion MAP abgebildet. Dann kann durch die Verwendung von CPU-Befehlen (MVC, CLI,...) auf die Daten einer Datei zugegriffen werden.

EAM – Evanescent Access Method

EAM ist die Zugriffsmethode für EAM-Dateien im BS2000. EAM-Dateien werden nicht katalogisiert. Das Eröffnen einer EAM-Datei erfordert deshalb keinen Plattenzugriff. Eine EAM-Datei wird bei Taskende automatisch gelöscht.

16.3 Zugriffsarten

- Blockorientierter Zugriff
- Satzorientierter Zugriff

16.3.1 Blockorientierter Zugriff

Das Benutzerprogramm verarbeitet die Datei blockweise (z.B. mit der Zugriffsmethode UPAM). Daher ist auch kein Blocken oder Entblocken von Sätzen nötig. Die Verarbeitung kann beschleunigt werden durch:

- Wechselpufferbetrieb/überlappende Ein-/Ausgabe

  Während das Programm den Inhalt eines Puffers bearbeitet, führt das DVS bereits die nächste Ein-/Ausgabeoperation durch. Bearbeitet z.B. das Programm (lesend) den Puffer A, kann das DVS bereits im Puffer B den nächsten Datenblock einlesen, sodass das Programm bei Abschluss der Bearbeitung vom Puffer A nur auf den Puffer B zu wechseln braucht usw.

- gekettete Ein-/Ausgabe.

  Das DVS überträgt bis zu 16 PAM-Seiten/Blöcke gleichzeitig zwischen Peripherie- und Hauptspeicher; Zeiteinsparung ergibt sich durch die geringere Zahl an Ein-/Ausgabeoperationen.

16.3.2 Satzorientierter Zugriff

Basis der logischen Dateiverarbeitung ist der Datensatz. Auch bei satzweisem Zugriff werden Datenblöcke zwischen Peripherie und Pufferbereich übertragen. Das DVS übernimmt anschließend das Blocken bzw. Entblocken der Sätze und stellt dem Benutzerprogramm die benötigten Sätze zur Verfügung. Wenn beim Lesen der gesuchte Satz nicht im Puffer enthalten ist oder beim Schreiben der Puffer gefüllt ist, wird eine Datenübertragung ausgelöst: das DVS liest den nächsten Datenblock bzw. schreibt den Pufferinhalt als neuen Datenblock.

Beim satzorientierten Zugriff müssen Übertragungs- und Ortungsbetrieb unterschieden werden.

Übertragungsbetrieb (Move Mode)

Im Programm ist ein Arbeitsbereich definiert. Das DVS überträgt beim Lesen den gesuchten Satz aus dem Puffer in den Arbeitsbereich; beim Schreiben überträgt es den Satz vom Arbeitsbereich in den Puffer. Das folgende Bild zeigt schematisch eine Schreiboperation.

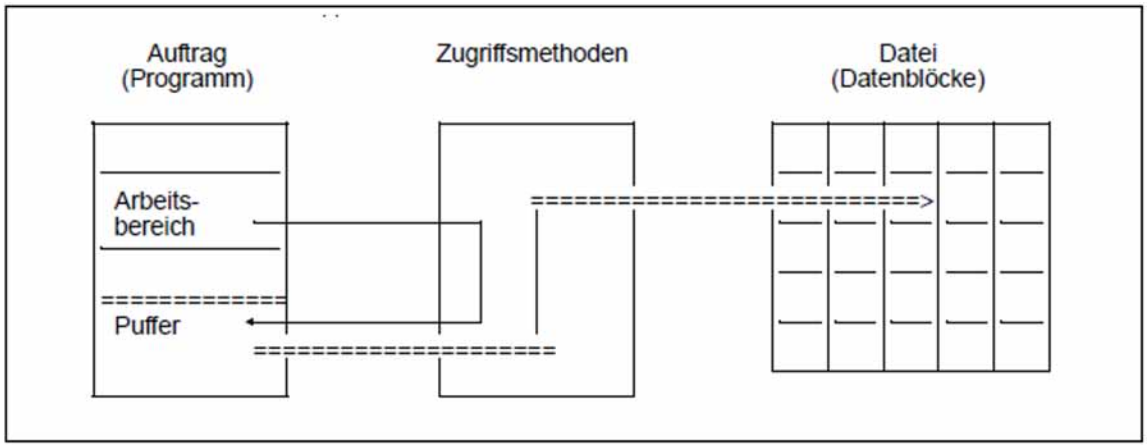


Bild 21: Übertragungsbetrieb bei satzorientiertem Zugriff

Ortungsbetrieb (Locate Mode)

Im Programm ist kein Arbeitsbereich definiert. Es werden keine Sätze zwischen Puffer und Arbeitsbereich übertragen, sondern die Sätze direkt im Puffer verarbeitet. Beim Lesen stellt das DVS dem Programm die Adresse des zu lesenden Satzes zur Verfügung, beim Schreiben übergibt es dem Programm die Adresse, an die der neue Satz geschrieben werden soll. Das folgende Bild zeigt schematisch eine Schreiboperation. Ein Pointer zeigt auf die Adresse im Puffer, an der der Satz erzeugt werden soll.

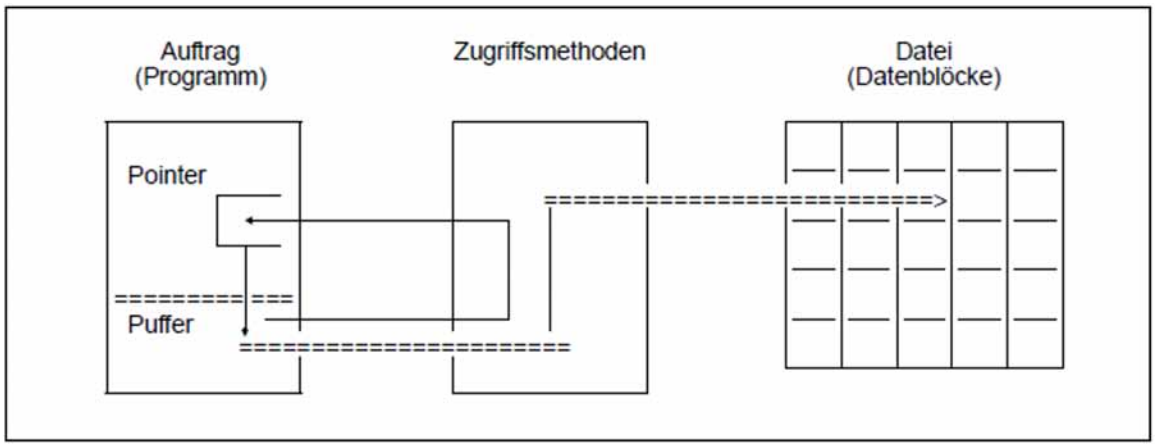


Bild 22: Ortungsbetrieb bei satzorientiertem Zugriff

16.4 Datei- und Verarbeitungseigenschaften

Bei allen Zugriffsmethoden werden die Dateieigenschaften festgelegt, wenn eine Datei erzeugt wird, d.h. wenn sie als Ausgabedatei eröffnet wird (OPEN OUTPUT/OUTIN).
Die Dateieigenschaften werden in den Katalogeintrag übernommen und gelten für alle folgenden Verarbeitungen. Sie können sowohl im Kommando ADD-FILE-LINK mit dem Operanden LINK-NAME als auch in der Dateierklärung des Programms definiert werden.

Soll eine bereits existierende Datei verarbeitet werden, kann im Kommando ADD-FILE-LINK die Nulloperanden-Funktion (Operandenwert *BY-CATALOG) ausgenutzt werden (siehe auch "Zugriff über den Dateikettungsnamen (Link-Name)" und "Informationen für die OPEN-Verarbeitung").

Im Gegensatz zu den Dateieigenschaften werden die Verarbeitungseigenschaften bei jeder Dateieröffnung neu festgelegt. Auch sie werden durch Operanden des Kommandos ADD-FILE-LINK und durch die Dateierklärung im Programm definiert.

Nulldateien

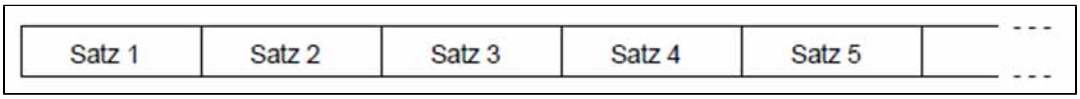
Eine Nulldatei entsteht, wenn eine Datei als Ausgabedatei (OPEN OUTPUT/OUTIN) geöffnet und anschließend sofort geschlossen wird. Wird für eine Nulldatei eine Leseoperation angestoßen, tritt das Ereignis „Dateiende“ ein, bei ISAM-Dateien kann auch das Ereignis „Schlüssel nicht vorhanden“ eintreten.

16.4.1 Satzformate

Das DVS unterscheidet drei Satzformate:

Satzformat F	mit Sätzen fester Länge
Satzformat V	mit Sätzen variabler Länge (Standard-Satzformat)
Satzformat U	mit Sätzen undefinierter Länge

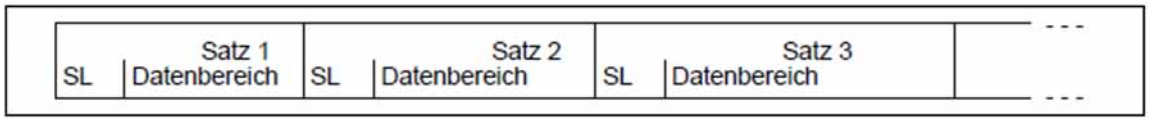
Satzformat F (Sätze fester Länge)



Eine Datei erhält die Eigenschaft „Sätze fester Länge“, wenn sie mit der Angabe RECORD-FORMAT=*FIXED im Kommando ADD-FILE-LINK oder entsprechender Angabe im Programm erstellt wird. Die Satzlänge ist mit dem Operanden RECORD-SIZE festzulegen.

Alle Sätze einer Datei haben die gleiche Länge, die im Katalogeintrag für die Datei eingetragen wird. Da die Satzlänge dem System bekannt ist, braucht der Benutzer bei Ein- oder Ausgabe kein Satzlängen- und Steuerfeld (s. u.) zu berücksichtigen.

Satzformat V (Sätze variabler Länge)

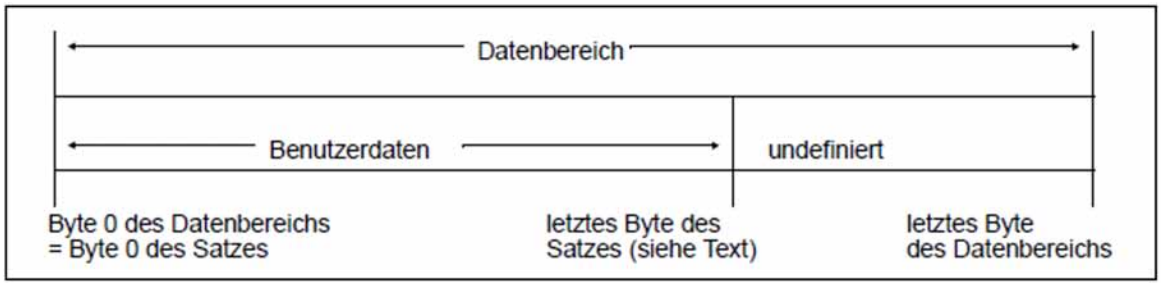


SL: Satzlänge = Länge des Datenbereichs + 4

Eine Datei erhält die Eigenschaft „Sätze variabler Länge“, wenn sie mit RECORD-FORMAT=*VARIABLE im Kommando ADD-FILE-LINK erstellt wird; RECORD-FORMAT= *VARIABLE ist Standard, wenn keine Angabe zum Satzformat getroffen wurde.

Bei Sätzen variabler Länge ist dem Datenbereich immer ein 4 Byte langes Steuerfeld vorgeschaltet, dessen erste beiden Byte binär die Länge des aktuellen Satzes inklusive Steuerfeld enthalten. Die beiden folgenden Byte sind reserviert.

Satzformat U (Sätze undefinierter Länge)



Eine Datei besteht aus „Sätzen undefinierter Länge“, wenn sie mit der Angabe RECORD-FORMAT=*UNDEFINED im Kommando ADD-FILE-LINK erstellt wird.

Dateien vom Format U enthalten pro Datenblock genau einen Satz. Die aktuelle Satzlänge, die der Benutzer beim Schreiben jeweils in einem Mehrzweckregister angeben muss, ist damit gleichzeitig auch die Blocklänge. Weil BS2000 die Blocklänge automatisch außerhalb des Datenbereichs in den blockspezifischen Verwaltungsinformationen (siehe z.B. Abschnitt "Blockformate für Plattendateien") ablegt, ist ein separates Speichern der Satzlänge nicht nötig. Wie bei fester Satzlänge muss der Benutzer also keine Satzlängen- oder Steuerfelder berücksichtigen: Ein Satz enthält nur die Nettodaten.

16.4.2 Blockformate für Plattendateien

Durch Weiterentwicklung der Plattenarchitektur wurden verschiedene Plattenformate eingeführt. Zur Unterstützung dieser Plattenformate wurden entsprechende Dateiformate geschaffen. Bei der Festlegung des Blockformats für Plattendateien ist das Plattenformat mit zu berücksichtigen.

16.4.2.1 Plattenformate

Zur Erhöhung der Netto-Speicherkapazität und zur Erhöhung der effektiven Datenübertragungsgeschwindigkeit wurden neue Plattenformate eingeführt. Folgende drei Kriterien eines Plattenformats sind dabei für das DVS relevant:

- PAM-Key (PAM-Schlüssel)
  Platte mit PAM-Key (K-Platte) / Platte ohne PAM-Key (NK-Platte)
- minimale Allokierungseinheit (min. AU) (Mindestgröße einer Datei)
  6 KB / 8 KB / 64 KB
- minimale Übertragungseinheit zwischen Platte und Hauptspeicher
  2 KB / 4 KB

Aus DVS-Sicht sind folgende Plattenformate zu unterscheiden:

- K-Platte
- NK-Platte
- NK2-Platte
- NK4-Platte

> ¡ Ein Net-Storage-Volume verhält sich in Bezug auf die Speicherplatzbelegung wie eine NK2-Platte mit minimaler Allokierungseinheit 8 KB.

K-Platte

Zu jedem 2-KB-Datenblock auf der Platte existiert ein außerhalb des Datenblocks liegendes Feld zur Aufnahme blockspezifischer Verwaltungsinformation, der sog. PAM-Key (16 Byte).
Die im PAM-Key hinterlegten Informationen werden (u.a.) von den DVS-Zugriffsarten SAM und ISAM ausgewertet; die Zugriffsmethode UPAM benötigt sie nicht (sofern nicht der Benutzer selbst auf den Benutzerteil des PAM-Key zugreift).
Die min. AU beträgt 6 KB. Auf K-Platten können Dateien aller Formate residieren.
Die K-Platte wird in allen bisher freigegebenen BS2000-Versionen unterstützt.

NK-Platte

Die Platte ist ohne PAM-Key formatiert (Non-Key-Platte). Die blockspezifische Verwaltungsinformation muss entweder im Datenblock geführt werden oder entfällt ganz. NK-Platte ist der Oberbegriff zu NK2- und NK4-Platte.

NK2-Platte

Die minimale Übertragungseinheit zwischen Platte und Hauptspeicher beträgt 2 KB. Durch einen Schreib/Lese-Auftrag werden also immer 2 KB oder Vielfache von 2 KB bewegt. Bei den NK2-Platten ist entsprechend der min. AU zu unterscheiden.
Es gibt Platten mit 6 KB und Platten mit 8 bzw. 64 KB min. AU.
Auf den Platten mit 8 bzw. 64 KB min. AU können sowohl NK2- als auch NK4-Dateien residieren.

NK4-Platte

Die minimale Übertragungseinheit zwischen Platte und Hauptspeicher beträgt 4 KB. Durch einen Schreib/Lese-Auftrag werden also 4 KB oder ein Vielfaches von 4 KB bewegt.
Die min. AU beträgt 8 KB oder 64 KB. Auf NK4-Platten können nur NK4-Dateien residieren.

Die nachstehende Tabelle zeigt die verschiedenen Plattenbezeichnungen.

Plattenbezeichnung	PAM-Key K = Key NK= Non-Key	Minimale Allokierungseinheit (min.AU)	Minimale Übertragungseinheit (min.TU)
K-Platte	K	6 KB	2 KB
NK2-Platte (6KB) [1)] = NK2-Platte mit min.AU 6 KB	NK	6 KB	2 KB
NK2-Platte (8KB) = NK2-Platte mit min.AU 8 KB	NK	8 KB	2 KB
NK2-Platte (64KB) = NK2-Platte mit min.AU 64 KB	NK	64 KB	2 KB
NK4-Platte (8KB) = NK4-Platte mit min.AU 8 KB	NK	8 KB	4 KB
NK4-Platte (64KB) = NK4-Platte mit min.AU 64 KB	NK	64 KB	4 KB

Tabelle 38: Übersicht über die verschiedenen Plattenformate/Pattenbezeichnungen

1) Bei der Plattenbezeichnung wird die minimale Allokierungseinheit nur angegeben, falls es für die Beschreibung des Sachverhalts relevant ist; ansonsten werden die Begriffe NK2-Platte und NK4-Platte verwendet.

Hinweise

- Das Plattenformat wird durch die Systembetreuung festgelegt.
- Das Plattenformat in einem SF-Pubset oder Volume-Set ist homogen. Zu jedem Plattenformat gibt es ein entsprechendes Pubset-Format.
- Mit dem Kommando SHOW-MASTER-CATALOG-ENTRY bzw. auf Programmebene mit dem Makro STAMCE kann man sich über die Formatierung seines Pubsets informieren.
- Für Privatplatten werden nur die Formate K- und NK2-Platte mit einer minimalen Allokierungseinheit von 6 KB unterstützt.

16.4.2.2 Dateiformate

Für die Darstellung der verschiedenen Block- und Dateiformate sind die folgenden Begriffe zu unterscheiden:

- PAM-Block
- logischer Block (Datenblock)
- Übertragungseinheit

PAM-Block:
Ein PAM-Block hat die Größe 2048 Byte + PAM-Key von 16 Byte. Nach Einführung des NK-Plattenformats hat es sich eingebürgert, unter einem PAM-Block auch eine Größe von 2048 Byte ohne PAM-Key zu verstehen. Ein PAM-Block wird auch als PAM-Seite oder 2-KB-Datenblock bezeichnet.

Logischer Block:
Ein logischer Block stellt einen zusammenhängenden Bereich von Daten/Sätzen dar. Seine Größe wird definiert im Kommando ADD-FILE-LINK, Operand BUFFER-LENGTH bzw. im Makro FILE, Operand BLKSIZE. Die Angabe kann durch *STD(SIZE=n) bzw. (STD,n) oder durch eine bestimmte Zahl n von Byte gegeben sein.
Bei der Angabe von (STD,n) wird bestimmt, dass ein logischer Block aus n 2048-Byte-Einheiten aufgebaut ist. (Soll eine Datei auf einer NK4-Platte allokiert werden, darf n nur eine gerade Zahl sein.)
Auf Bändern und Magnetbandkassetten ist daneben noch die explizite Längenangabe in Byte möglich. Allerdings stellt diese Angabe lediglich die maximale Länge eines logischen Blocks dar. Die tatsächliche Länge – kleiner gleich n-Byte – ergibt sich dann durch die Daten selbst.

Übertragungseinheit:
Die Übertragungseinheit gibt die Länge der Daten an, die mit einer physikalischen Ein-/Ausgabe zwischen Hauptspeicher und Platte transportiert werden.

Bei Platten ist die Übertragungseinheit gleich der Länge eines logischen Blocks. Bei NK2-Platten ist die Übertragungseinheit ein Vielfaches von 2 KB (kleinste Übertragungseinheit ist 2 KB), bei NK4-Platten ist die Übertragungseinheit ein Vielfaches von 4 KB (kleinste Übertragungseinheit ist 4 KB).

Bei Bändern sind zwei Fälle zu unterscheiden:

- Die Blocklänge einer Datei ist mit BLKSIZE=(STD,n) im Makro bzw. mit BUFFER-LENGTH=*STD(SIZE=n) im Kommando definiert. Damit wird für Banddateien eine logische Blocklänge von n * 2 KB festgelegt. Die Übertragungseinheit ist in diesem Fall 2 KB. Es wird n-mal eine Einheit von 2048 Byte transportiert.
  Es werden n Ein-/Ausgaben von je 2 KB durchgeführt.
- Die Blocklänge einer Datei ist mit BLKSIZE=n im Makro bzw. mit BUFFER-LENGTH=n im Kommando definiert. In diesem wird für Banddateien eine logische Blockgröße von n Byte festgelegt. Es wird eine Ein-/Ausgabe von n Byte durchgeführt. In diesem Fall stimmt auch bei Banddateien die logische Blockgröße mit der Übertragungseinheit überein.

Zur Unterstützung der verschiedenen Plattentypen bietet das DVS verschiedene Datei- und Blockformate an. Die beiden grundsätzlichen Blockformate sind:

- Blockformat mit PAM-Schlüsseln
  im Folgenden als K-Dateiformat (für Key-Dateiformat) bezeichnet
- Blockformate ohne PAM-Schlüssel
  im Folgenden als NK-Dateiformate (für Non-Key-Dateiformate) bezeichnet

K-Dateiformat

Die block-spezifische Verwaltungsinformation (Blockkontroll-Information) wird im PAM-Schlüssel (16 Byte) geführt. 8 Byte des PAM-Schlüssels werden durch das System genutzt, weitere 8 Byte werden durch die höheren Zugriffsmethoden SAM und ISAM genutzt. Lediglich mit UPAM kann man über diesen „User“-Teil des PAM-Schlüssels verfügen.
Eine Nutzung dieses Teils des PAM-Schlüssels sollte vermieden werden, um eine Umstellung der Anwendung auf ein NK-Format zu vereinfachen.

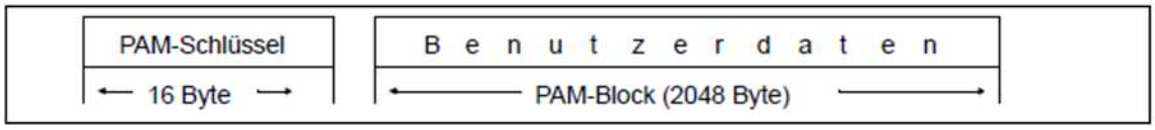
Dieses Blockformat wird von allen Zugriffsmethoden außer DIV und FASTPAM unterstützt. Eine Datei kann im K-Format erstellt werden, sofern der Datenträger die Aufzeichnung des PAM-Schlüssels erlaubt (K-Pubset bzw. K-Privatplatte).

Nähere Informationen können den Abschnitten über die einzelnen Zugriffsmethoden entnommen werden.

*BLOCK-CONTROL-INFO=*PAMKEY*

Mit dem Kommando ADD-FILE-LINK, Operand BLOCK-CONTROL-INFO=*PAMKEY bzw. mit dem Makro FILE, Operand BLKCTRL=PAMKEY wird ein K-Format festgelegt.

Das folgende Bild veranschaulicht einen PAM-Block mit dem dazugehörigen PAM-Schlüssel.



NK-Dateiformate

Bei den NK-Formaten sind zwei Dateiformate zu unterscheiden, bei denen die block-spezifische Verwaltungsinformation innerhalb eines jeden Datenblockes liegt (Operand BLOCK-CONTROL-INFO=*WITHIN-DATA-2K-BLOCK/*WITHIN-DATA-4K-BLOCK im Kommando ADD-FILE-LINK; die Angabe *WITHIN-DATA-BLOCK umfasst 2K und 4K). Dieser Bereich steht für Benutzerdaten nicht zur Verfügung.
Darüber hinaus gibt es die Möglichkeit, ein NK-Dateiformat zu wählen, bei dem keine blockspezifische Verwaltungsinformation hinterlegt wird: BLOCK-CONTROL-INFO=*NO.

Bei den Zugriffsmethoden UPAM und SAM ist die block-spezifische Verwaltungsinformation immer 16 Byte lang (12 Byte Block-Kontrollfeld und 4 Byte Datenlängenfeld).

Bei einer NK2-ISAM-Datei (BLOCK-CONTROL-INFO=*WITHIN-DATA-2K-BLOCK) ist die block-spezifische Verwaltungsinformation in den ersten 16 Byte eines jeden 2-KB-Blockes hinterlegt.
Bei einer NK4-ISAM-Datei (BLOCK-CONTROL-INFO=*WITHIN-DATA-4K-BLOCK) ist die block-spezifische Verwaltungsinformation in den ersten 16 Byte eines jeden 4-KB-Blockes hinterlegt.

Bei der Konvertierung einer K-Datei zu einer NK-Datei ist Folgendes zu beachten:
Wird die maximale Satzlänge (16 PAM-Blöcke = 32 KB) ausgeschöpft, kommt es zu Inkompatibilitäten, da kein Platz mehr für die block-spezifische Verwaltungsinformation zur Verfügung steht.

Auf einer NK4-Platte können nur NK4-Dateien liegen.

Nähere Informationen können den Abschnitten über die einzelnen Zugriffsmethoden entnommen werden.

*BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK*

Mit dem Kommando ADD-FILE-LINK, Operand BLOCK-CONTROL-INFO= *WITHIN-DATA-BLOCK bzw. mit dem Makro FILE, Operand BLKCTRL=DATA, wird ein NK-Format festgelegt, wobei beim Anlegen der Datei eine NK2- oder eine NK4-Datei erstellt wird:

Beim Anlegen einer Datei wird folgendermaßen vorgegangen:

PAM- und SAM-Dateien:

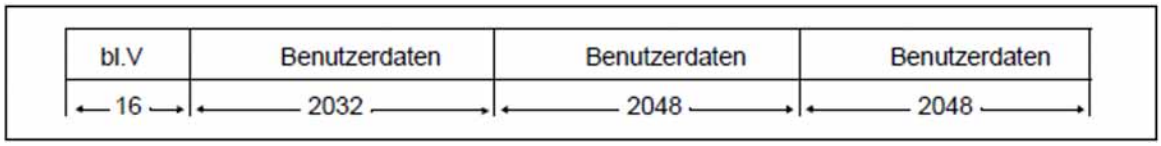
Ist der Blockungsfaktor n eine ungerade Zahl, wird eine NK2-Datei angelegt. Ist der Blockungsfaktor n eine gerade Zahl, wird eine NK4-Datei angelegt.

ISAM-Datei:

Ist das Blockformat der ISAM-Datei nicht explizit angegeben (siehe unten), wird abhängig vom Plattenformat, auf dem die ISAM-Datei angelegt wird, eine NK2-ISAM-Datei oder eine NK4-ISAM-Datei angelegt.

Eine bereits bestehende Datei kann unabhängig vom Blockformat geöffnet werden.

Das folgende Bild veranschaulicht einen logischen Datenblock einer NK2-SAM-Datei (Länge 6 KB, Blockungsfaktor n=3; bl.V = block-spezifische Verwaltungsinformation).

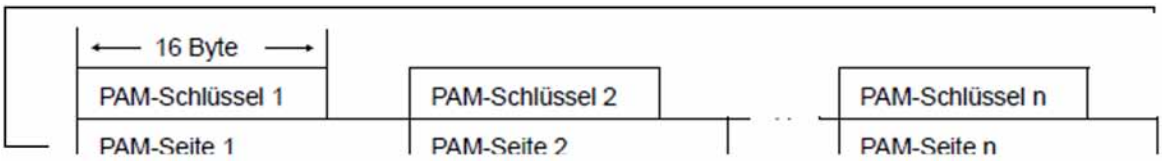


*BLOCK-CONTROL-INFO=*WITHIN-DATA-2K-BLOCK (für NK-ISAM-Dateien)*

Mit dem Kommando ADD-FILE-LINK, Operand BLOCK-CONTROL-INFO=*WITHIN-DATA-2K-BLOCK bzw. dem Makro, FILE Operand BLKCTRL=DATA2K, wird explizit eine NK2-Datei erstellt (die Angabe kann auch für SAM und PAM erfolgen; sie wirkt dann wie *WITHIN-DATA-BLOCK).

Die block-spezifischen Verwaltungsinformationen sind in den ersten 16 Byte eines jeden 2-KB-Blockes hinterlegt.

Das folgende Bild veranschaulicht einen logischen Datenblock einer NK2-ISAM-Datei (Länge 6 KB, Blockungsfaktor n=3; bl.V = block-spezifische Verwaltungsinformation).



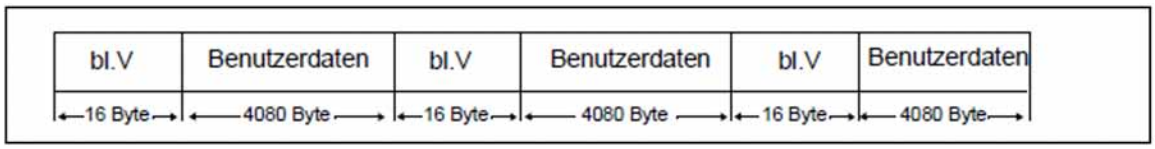
*BLOCK-CONTROL-INFO=*WITHIN-DATA-4K-BLOCK (für NK-ISAM-Dateien)*

Mit dem Kommando ADD-FILE-LINK, Operand BLOCK-CONTROL-INFO=*WITHIN-DATA-4K-BLOCK bzw. mit dem Makro FILE, Operand BLKCTRL=DATA4K, wird explizit eine NK4-Datei angelegt (die Angabe kann auch für SAM und PAM erfolgen; sie wirkt dann wie *WITHIN-DATA-BLOCK).

Die block-spezifische Verwaltungsinformation wird in den ersten 16 Byte eines jeden 4-KB-Blockes hinterlegt.

Eine NK4-ISAM-Datei kann auf beliebigen Platten erstellt werden.

Das folgende Bild veranschaulicht einen logischen Datenblock einer NK4-ISAM-Datei (Länge 12 KB, Blockungsfaktor n=6; bl.V = block-spezifische Verwaltungsinformation).



*BLOCK-CONTROL-INFO=*NO*

Mit dem Kommando ADD-FILE-LINK, Operand BLOCK-CONTROL-INFO=*NO bzw. mit dem Makro FILE, Operand BLKCTRL=NO, wird ein NK-Format festgelegt.

Das System hinterlegt keine block-spezifischen Verwaltungsinformationen, weder im PAM-Schlüssel, noch innerhalb des Datenblocks.

Dieses Format existiert nur für UPAM-Plattendateien, UPAM-Banddateien und SAM-Banddateien.

Bei SAM-Plattendateien und ISAM-Plattendateien wird die Angabe BLOCK-CONTROL- INFO=*NO bzw. BLKCTRL=NO intern in BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK bzw. BLKCTRL=DATA umgewandelt.

Nähere Informationen können den Abschnitten über die einzelnen Zugriffsmethoden entnommen werden.

Das folgende Bild veranschaulicht einen logischen Datenblock einer NK2-PAM-Datei ohne block-spezifische Verwaltungsinformation (Länge 6 KB, Blockungfaktor n=3).

Zur Unterstützung der verschiedenen Plattentypen bietet das DVS verschiedene Datei- und Blockformate an. Bei der Festlegung des Dateiformats ist die Abhängigkeit der folgenden Dateieigenschaften voneinander zu beachten:

- Dateistruktur
- Blockstruktur
- logische Blocklänge (Blockungsfaktor)

Tabelle der Kombinationsmöglichkeiten von Dateiformat und anderen Dateieigenschaften:

Dateiformat	Dateistruktur (ACCESS-METHOD)	Blockstruktur (BLOCK-CONTROL-INFO)	logische Blocklänge (BUFFER-LENGTH)	Blockungs-Faktor (n)	Platten-Format
K-Datei	PAM	PAMKEY			
	SAM				
	ISAM				
NK2-Datei	PAM	NO oder DATA	STD	ungerade	K-Platte
	SAM	DATA	STD	ungerade	
	ISAM	DATA2K			

		DATA			NK2-Platte (6KB/8KB /64KB)
NK4-Datei	PAM	NO oder DATA	STD	gerade	K-Platte
	SAM	DATA	STD	gerade	NK2-Platte (6KB/8KB /64KB)
	ISAM	DATA4K	STD	gerade	NK4-Platte

Tabelle 39: Übersicht über die verschiedenen Dateiformate

Die minimale Allokierungseinheit wirkt sich nicht auf das Dateiformat aus.

Auf NK4-Platten können nur Dateien mit einem geradzahligen Blockungsfaktor abgelegt werden.

Das K-Dateiformat wird in allen bisher freigegebenen BS2000-Versionen unterstützt.

16.4.3 Blockformate für Banddateien

Ein physikalischer Block ist die Einheit der Datenübertragung zu oder von einem Ein-/Ausgabegerät. Die Daten zwischen 2 Blocklücken eines Magnetbandes bilden einen physikalischen Block. Ein solcher Block reicht nie über die Bandgrenze hinaus. Bei Banddateien wird unterschieden zwischen Standard-Blockformat und Nicht-Standard-Blockformat.
Eine Banddatei wird grundsätzlich mit dem Gerätetreiber PTAM verarbeitet.

Standard-Blockformat

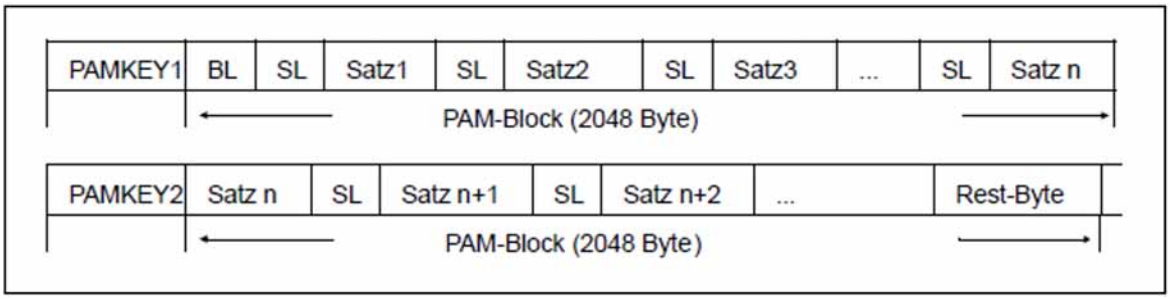
*BLOCK-CONTROL-INFO=*PAMKEY, BUFFER-LENGTH=*STD(SIZE=n)*

Ein Standardblock entspricht n PAM-Seiten. Der PAM-Schlüssel wird durch die Datenkettungseinrichtung der PAM-Seite vorangestellt und steht nicht gesondert auf dem Band. Als Treiber für Bänder steht nur PTAM zur Verfügung.

Die Angabe BUFFER-LENGTH bestimmt, aus wie vielen PAM-Seiten ein Standardblock besteht. Bei der Zugriffsmethode SAM ist ein Standardblock nicht immer vollständig mit logischen Sätzen belegt. Der Platz zwischen Ende des letzten logischen Satzes und dem Ende des Blocks hat undefinierten Inhalt (siehe „Rest-Byte“ im Beispiel unten).
Eine gekettete Ein-/Ausgabe ist auf Bändern nicht möglich.

Beispiel

```
BUFFER-LENGTH=*STD(SIZE=2), RECORD-FORMAT=*VARIABLE
```



Nicht-Standard-Blockformate

Ein Nicht-Standardblock ist ein Datenblock auf Magnetband, dem kein PAM-Schlüssel vorangestellt ist. Nicht-Standardblöcke können höchstens so lang sein, wie im Kommando ADD-FILE-LINK beim Operanden BUFFER-LENGTH angegeben ist (max. 32768 Byte). Nicht-Standardblöcke werden immer dann erzeugt, wenn BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK oder *NO angebeben wurde, auch in Verbindung mit der Angabe BUFFER-LENGTH=*STD(SIZE=n).
Für eine Bearbeitung mit UPAM muss BUFFER-LENGTH ein Vielfaches von 2048 betragen.

*Datenblock mit Verwaltungsinformation (BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK)*

Dem Block wird kein PAM-Schlüssel vorgeschaltet. Stattdessen stellen SAM und UPAM an den Beginn eines jeden Logischen Blockes ein 12 Byte langes Blockkontrollfeld (Control Field, CF), an das SAM ein 4 Byte langes Datenlängenfeld (Data Length, DL) anfügt. Diese 12 Byte (CF bei UPAM) bzw. 16 Byte (CF und DL bei SAM) werden als Pufferverschiebung behandelt.

Beispiel

```
BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK,BUFFER-LENGTH=*STD(SIZE=2),
RECORD-FORMAT=*VARIABLE, ACCESS-METHOD=*SAM
```

*Datenblock ohne Verwaltungsinformation (BLOCK-CONTROL-INFO=*NO)*

Dieser Block enthält keine block-spezifische Verwaltungsinformation. Dieses Format existiert für UPAM- und SAM-Dateien.

Beispiel

```
BLOCK-CONTROL-INFO=*NO,BUFFER-LENGTH=4096,RECORD-FORMAT=*VARIABLE,
ACCESS-METHOD=*SAM, Datenlänge 3900 Byte
```

Auf das Band werden nur die Daten geschrieben (hier 3900 Byte).

Innerhalb einer Datei können die Datenblöcke unterschiedlich lang sein. Es kann jedoch kein Block die im Operanden BUFFER-LENGTH des Kommandos ADD-FILE-LINK definierte maximale Blocklänge überschreiten.

Für SAM-Dateien gilt:

Das Datenlängenfeld (DL) ist lt. DIN-Norm nicht Bestandteil des Satzes, bei Sätzen vom Format V (RECORD-FORMAT=*VARIABLE) entsteht deshalb eine Pufferverschiebung (BLOCK-OFFSET) um vier Byte.
Geben Sie beim Kommando ADD-FILE-LINK den Operanden BLOCK-OFFSET=0 an, erfolgt keine Pufferverschiebung, d.h., es wird kein Blocklängenfeld berücksichtigt.

Blockverlängerung durch Füllzeichen wird von der Zugriffsmethode SAM und von der Kennsatzbehandlung beim Lesen unterstützt (z.B. Lesen von Nicht-BS2000-Bändern).

Aufbau von Daten- und Satzlängenfeld (nur bei SAM-Dateien)

im EBCDI-Code:	2 Byte lange Binärzahl, die die Block- bzw. Satzlänge enthält, gefolgt von zwei Leerzeichen (2X'40')
im ISO-Code:	eine vierstellige Dezimalzahl (entspricht Format D)

Da im Format D die Dezimalzahl maximal vierstellig ist, ist die maximale Blocklänge 9999 Byte.

Dieses Blockformat ist das Blockformat für den Datenaustausch nach DIN 66029 und wird auf- und abwärts kompatibel bleiben.

Beispiel: SAM-Datei mit Sätzen variabler Länge und Nicht-Standardblöcken

Der Block (Blocklänge = 198 Byte) enthält 3 Sätze von 76, 84 bzw. 34 Byte Länge. Block- und Satzlängenfelder (BL/SL) haben folgenden Inhalt:

	BL	SL-1	SL-2	SL-3
EBCDI-Code (Format V)	00C64040	004C4040	00544040	00224040
ISO-Code (Format D)	F0F1F9F8	F0F0F7F6	F0F0F8F4	F0F0F3F4

Zugriffsmethoden und Blockformat bei Banddateien

Das Blockformat (Standard- oder Nicht-Standardblock) wird durch die Angabe bei den Operanden BLOCK-CONTROL-INFO, BUFFER-LENGTH oder LABEL-TYPE (Blocklänge bzw. Band-Normtyp) und die Zugriffsmethode bestimmt. Die folgende Tabelle zeigt die zulässigen BLOCK-CONTROL-INFO-Werte in Abhängigkeit von der Zugriffsmethode:

BLOCK-CONTROL-INFO	UPAM	SAM	BTAM
*PAMKEY	+	+	
*WITHIN-DATA-BLOCK	+	+	
*NO	+	+	+

Tabelle 40: Banddateien: Blockformat / Zugriffsmethoden

Der vorhergehenden Tabelle ist zu entnehmen, dass BTAM nur Dateien mit Nicht-Standardblöcken (BLOCK-CONTROL-INFO=*NO) verarbeiten kann. Mit den Zugriffsmethoden
SAM und UPAM können Dateien aller Blockformate (Standard- und Nicht-Standardblöcke) verarbeitet werden (BLOCK-CONTROL-INFO=*PAMKEY/*WITHIN-DATA-BLOCK/*NO).

Eine Datei kann nicht gleichzeitig Standard- und Nicht-Standardblöcke enthalten.

16.4.4 Zusammenhang Satz / Block / PAM-Seite

Ein Datensatz ist aus der Sicht des DVS ein nicht weiter zerlegbares Datenelement. Dies gilt sowohl für das Auffinden in einer Datei als auch für die Übertragung vom bzw. zum aufrufenden Programm.

Daraus ergibt sich, dass ein Datenblock mindestens so lang sein muss wie der längste Satz der Datei. Bei Plattendateien muss das Blockformat mit dem des PPAM verträglich sein.

Ein Datenblock besteht aus einer oder mehreren PAM-Seiten. Banddateien können aus Standard- oder aus Nicht-Standardblöcken bestehen. Die maximale Datenblocklänge ist für Platten- und Banddateien 32768 Byte.

Beispiele zum Verhältnis Satz/Datenblock/PAM-Seite bei SAM

1. Dateieigenschaften:

`RECORD-FORMAT=*FIXED`	Sätze fester Länge
`RECORD-SIZE=100`	Satzlänge 100 Byte
`BUFFER-LENGTH=*STD(SIZE=2)`	Blocklänge = 2 PAM-Seiten = 2*2048 Byte = 4096 Byte

   Jeder Datenblock enthält 40 Sätze, es bleiben also jeweils 96 Byte frei. Gleichzeitig besteht jeder Datenblock aus 2 PAM-Seiten. Das DVS speichert die Datensätze innerhalb eines Datenblocks ohne Rücksicht auf Grenzen zwischen PAM-Seiten. In diesem Beispiel sind die ersten 48 Byte von Satz 21 in der ersten PAM-Seite, die restlichen 52 Byte in der zweiten PAM-Seite enthalten.

2. Dateieigenschaften:

`RECORD-FORMAT=*FIXED`	Sätze fester Länge
`RECORD-SIZE=1500`	Satzlänge 1500 Byte
`BUFFER-LENGTH=*STD`	Blocklänge = 1 PAM-Seite

   Diese Kombination ist sehr ungünstig, da in jedem Block 548 Byte „verschwendet“ werden. Eine erstrebenswerte Blocklänge wären 6144 Byte (3 PAM-Seiten, BUFFER-LENGTH=*STD(SIZE=3), dann könnte der Datenblock 4 Sätze aufnehmen, und von den 6144 Byte würden 6000 genutzt.

3. Dateieigenschaften:

`RECORD-FORMAT=*FIXED`	Sätze fester Länge
`RECORD-SIZE=8192`	Satzlänge 8192 Byte
`BUFFER-LENGTH=*STD(SIZE=8)`	Blocklänge = 8 PAM-Seiten

   Jeder Datenblock enthält 2 Sätze, jeder Datensatz benötigt 4 PAM-Seiten.

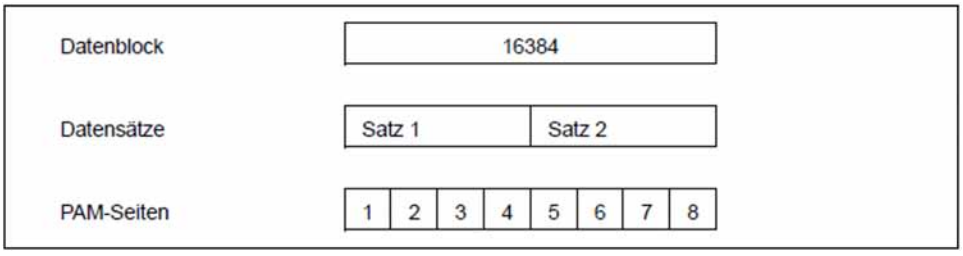


Bild 23: Datenblock/PAM-Seite/Satz

17 BTAM - Basic Tape Access Method

BTAM bietet Ihnen ein effektives und flexibles Mittel zum Speichern und Wiederauffinden von Blöcken in sequenziell organisierten Banddateien. Mit BTAM können Magnetbänder, Magnetbandkassetten (MBK), auch BS2000-fremde Bänder, verarbeitet werden.

Mit BTAM lassen sich auch Banddateien verarbeiten, die Nicht-Standardkennsätze oder keine Kennsätze haben oder die entsprechend DIN 66029, Austauschstufe 1, erstellt wurden (entspricht der Angabe LABEL=(STD,1) im FILE-/FCB-Makro bzw. der Angabe LABEL-TYPE=*STD(DIN-REVISION-NUMBER=1) im Kommando ADD-FILE-LINK).

Mit dem BTAM-Makroaufruf können Bänder beliebig block- oder abschnittsweise positioniert werden. Im Programm können Datenblöcke gelesen und/oder geschrieben oder die Verarbeitungsrichtung geändert werden, ohne dass eine CLOSE-Operation notwendig ist!

Von BTAM unsterstützte Codes sind:

- EBCDIC
- ISO7-Bit-Code
- OWN-Code

Wie bei der SAM-Bandverarbeitung ergeben sich Wechselwirkungen zwischen Kennsatzangabe und Code.

BTAM-Funktionen

- Gekettete Ein-/Ausgabe
  (Operand CHAINIO im Makro BTAM)

  spart CPU-Leistung und Zeit für Daten-Ein-/Ausgabe
- MAV-Modus
  (Multi-Auftragsverwaltung, Operanden BTAMRQS oder REQNO im Makro BTAM)

  Mehrere Ein-/Ausgaben auf Band können parallel (asynchron) verarbeitet werden. Da der MAV-Modus stark zeitabhängig ist, kann er nur mit geketteter Ein-Ausgabe sinnvoll eingesetzt werden: je größer der Kettungsfaktor, desto besser sind Geräteauslastung und Zeitverhalten. Das Zeitverhalten bei Ein-/Ausgabe wird dadurch verbessert, dass ständig Ein-/Ausgaben stattfinden können und das Band nicht immer wieder neu positioniert werden muss.
  Bei entsprechenden BTAM-Ein-/Ausgabeaufträgen wartet BTAM nicht auf das Ende der Ein-/Ausgabe, sondern gibt die Kontrolle sofort an den Aufrufer zurück. Der Aufrufer kann sich das Ergebnis der Ein-/Ausgabeoperation später mit einem WAIT/CHECK-Aufruf einholen. Zu welchem BTAM-Ein-/Ausgabeauftrag der spätere WAIT /CHECK gehört, muss jedem asynchronen BTAM-Auftrag mittels einer
  Request-Nummer (Parameter REQNO im Makro BTAM) mitgegeben werden.Der Performance-Gewinn wird dadurch erzielt, dass von BTAM mehrere asynchrone Aufträge gleichzeitig ausgeführt werden können. Mit dem nächsten Ein-/Ausgabeauftrag muss also nicht gewartet werden, bis der vorangehende komplett ausgeführt worden ist.
  Die höchste Request-Nummer, die vergeben werden darf (und die damit auch die maximale Anzahl der für einen Auftrag gleichzeitig laufenden asynchronen Ein-/Ausgaben darstellt), wird mit dem Parameter BTAMRQS festgelegt.
- Gepufferte Ein-/Ausgabe
  (Operand TAPEWR im Makro BTAM)

  für Dateien auf MBK

Makroaufrufe für BTAM

Zur Verarbeitung von Banddateien mit der Zugriffsmethode BTAM stehen folgende Makros zur Verfügung:

- alle Service-Makroaufrufe (OPEN, CLOSE usw.)
- der Makroaufruf BTAM mit folgenden Funktionen:

Funktion	Bedeutung
CHK	Verarbeitungszustand einer Ein-/Ausgabeoperation feststellen
ERG	Blocklücke erzeugen
POS	Band positionieren
RBID	Bandposition bestimmen
RD/RDWT	Daten in den Hauptspeicher lesen und den Abschluss der Ein-/Ausgabeoperation abwarten
RDBF	Daten aus dem Sicherstellungsbereich des MBK-Puffers lesen
REV/REVWT	Daten rückwärts lesen / und den Abschluss der Ein-/Ausgabeoperationabwarten
RT/RTL	Lesen mit Datentransfer; mit/ohne Meldung bei kürzerer Länge als erwartet
RNT/RNTL	Lesen ohne Datentransfer; mit/ohne Meldung bei kürzerer Länge als erwartet
SYNC	Synchronisieren und Bandposition bestimmen
WRT/WRTWT	Daten aus dem Hauptspeicher schreiben und den Abschluss der Ein-/Ausgabeoperation abwarten
WT	Warten auf Abschluss der Ein-/Ausgabeoperation

Tabelle 41: Funktionen des Makros BTAM

- Steuerschlüssel für Positionieren und für das Schreiben von Abschnittsmarken (im BTAM-Makroaufruf):
  BSF, BSR, FSF, FSR, REW, RUN, WTM

17.1 BTAM-Satz- und Blockformate

BTAM ist eine blockorientierte Zugriffsmethode, d.h. BTAM erkennt keine Satzstrukturen innerhalb eines Blocks. Die Angabe RECORD-FORMAT=*FIXED im Kommando ADD-FILE-LINK bzw. RECFORM=F im Makro FILE wertet BTAM dahingehend aus, dass alle Blöcke dieser Datei eine feste Blocklänge haben.

BTAM versorgt folgende Satzformate:

- F für Sätze fester Länge
- U für Sätze undefinierter Länge
- V für Sätze variabler Länge (Satzformat V wird behandelt wie Satzformat U)

BTAM verarbeitet nur Dateien des Formats BLKCTRL=NO bzw. BLOCK-CONTROL-INFO=*NO.

Gilt RECFORM=U bzw. RECORD-FORMAT=*UNDEFINED, entnimmt das DVS die Blocklänge

- dem Register, das in FILE/FCB mit RECSIZE=reg definiert wurde
- der Angabe im Aktionsmakroaufruf (Operand LEN)
- der Angabe des Operanden RECORD-SIZE im Kommando ADD-FILE-LINK

Gilt RECFORM=F bzw. RECORD-FORMAT=*FIXED, gilt die mit BLKSIZE bzw. mit RECORD-SIZE definierte Blocklänge oder die Angabe im Operanden LEN des BTAM-Makroaufrufs.

Für die Blocklänge gilt: 18 Byte <= Blocklänge <= 32768 Byte.

Dateien, die mit SAM erstellt wurden, können mit BTAM verarbeitet werden. Der Benutzer muss jedoch selbst für das Entblocken der Sätze sorgen.

17.2 BTAM-Datei eröffnen

Das Eröffnen einer BTAM-Datei ist abhängig vom OPEN-Modus. Die verschiedenen Modi und die daraus abgeleiteten Verarbeitungsmöglichkeiten für eine BTAM-Datei sind in der folgenden Tabelle zusammengestellt.

OPEN-Modus	Verarbeitung
INOUT	Auffinden von Sätzen in einer vorhandenen Datei sowie Hinzufügen von Sätzen; es werden keine Anfangskennsätze erstellt, da angenommen wird, dass die Datei bereits vorhanden ist
INPUT	Auffinden von Sätzen in einer vorhandenen Datei in Richtung Dateiende
OUTIN	Erstellen einer neuen Datei und/oder Auffinden von Sätzen in der Datei; es werden Kennsätze erstellt, da eine neue Datei eingerichtet wird
OUTPUT	Erstellen einer neuen Datei
REVERSE	Wie INPUT, das Band wird jedoch beim OPEN auf das Dateiende positioniert. Dateien, die sich über mehrere Datenträger erstrecken, lassen sich nur einzeln (mit VSEQ im Makro FILE bzw. mit VOL-SEQUENCE-NUMBER im Kommando ADD-FILE-LINK) verarbeiten
SINOUT	Wie INOUT, das Band wird jedoch nicht positioniert; die Angabe ist unzulässig, wenn auf Bandanfang positioniert ist

Tabelle 42: OPEN-Modi für BTAM-Dateien

Die OPEN-Modi INPUT und REVERSE unterscheiden sich zum OPEN-Zeitpunkt nur in der Bandpositionierung. Ein OPEN REVERSE mit anschließender RD- oder RDWT-Operation führt nicht zum Rückwärtslesen.

17.3 Dateien auf MBK

Die Bandoperationen während der OPEN-Verarbeitung erfolgen ungepuffert, ebenso während der EOV- und CLOSE-Verarbeitung.

Ist die OPEN-Verarbeitung beendet, wird der Ausgabemodus auf gepufferte Verarbeitung umgestellt, falls nicht TAPEWR=IMMEDIATE im FILE-Makroaufruf oder FCB bzw. der Operand TAPE-WRITE=*IMMEDIATE im Kommando ADD-FILE-LINK angegeben wurde.

Sobald für Ausgabedateien EOV-/EOF-Verarbeitung eingeleitet wird, veranlasst das DVS eine Synchronisation, um sicherzustellen, dass alle Benutzerdaten auf Band geschrieben wurden. Nach Entleeren des Puffers werden die Endekennsätze ungepuffert geschrieben.

Wird bei der automatischen Synchronisation am Beginn der EOV-/EOF- oder CLOSE-Verarbeitung ein nicht-behebbarer Schreibfehler bei einem der im MBK-Puffer stehenden Blöcke erkannt, wird der EXLST-Ausgang CLOSER aktiviert, der Benutzer kann auf die Daten im Puffer jedoch nicht mehr zugreifen. Er kann diese Hürde umgehen, wenn er mit dem SYNC-Befehl die Synchronisation explizit veranlasst. Nicht-behebbare Schreibfehler werden dann über den EXLST-Ausgang ERRADR gemeldet. Da die Daten im MBK-Puffer noch zugreifbar sind, kann der Schaden noch behoben werden.

18 DIV - Data In Virtual

Data in Virtual (DIV) ist eine Zugriffsmethode, die sich von den traditionellen Zugriffsmethoden wie ISAM, SAM oder UPAM dadurch unterscheidet, dass sie ohne eine Strukturierung der Datei in Sätze oder Blöcke, ohne I/O-Puffer und ohne Operationen wie GET oder PUT auskommt.
DIV ist eine objektorientierte Zugriffsmethode, insbesondere geeignet zur Bearbeitung unstrukturierter Daten, sog. BLOBS (Binary Large Objects).

Mit der DIV-Funktion MAP kann eine Datei einem Bereich in einem virtuellen Adressraum (Programmraum, Datenraum) zugeordnet werden. Der Bereich im virtuellen Adressraum bildet dann ein Fenster, in dem die Seiten der Datei erscheinen, wenn auf sie zugegriffen wird.

Auf die Daten kann mit CPU-Befehlen zugegriffen werden. In einem Fenster geänderte Daten können mit der DIV-Funktion SAVE in die Datei zurückgeschrieben werden.

Mit DIV können NK-PAM-Dateien bearbeitet werden, die keine Datenverwaltungsinformationen enthalten (BLOCK-CONTROL=NO). Der systemübergreifende Dateizugriff (RFA) wird nicht unterstützt.

Neben dem objektorientierten Zugriff kann DIV durch Einsparung von Plattenzugriffen erhebliche Performance-Gewinne bieten.

DIV eignet sich für Anwendungen, bei denen keine Strukturierung in Sätze oder Blöcke erforderlich ist. Ein hoher Performance-Gewinn kann sich ergeben, wenn wiederholt auf Daten zugegriffen wird, die bereits durch einen vorangegangenen Zugriff in ein Fenster eingelesen wurden.

Mit den Funktionen FCT=*MAP und FCT=*UNMAP des Makros DIV können Fenster für einen Programmraum auf- und abgebaut werden. Für den Operanden SPID (Kennung des Datenraumes) muss stets SPID=0 angegeben werden.

DIV-Funktionen

Die DIV-Funktionen werden durch den Aufruf des Makros DIV genutzt. Es gibt folgende Grundfunktionen:

Funktion	Bedeutung
OPEN	Datei öffnen
MAP	Fenster (Arbeitsbereich) im virtuellen Adressraum definieren
SAVE	Modifizierte Seiten aus dem Fenster in die Datei auf Platte zurückschreiben
RESET	Änderungen im Fenster zurücknehmen
UNMAP	Fenster im virtuellen Adressraum freigeben
CLOSE	Datei schließen; ggf. noch existierende Fenster werden mit Standardwerten freigegeben.

Tabelle 43: DIV-Makro (Funktionen)

Mit dem Adapter Window Services ist es in einigen höheren Programmiersprachen möglich, DIV-Funktionen aus Programmen über eine CALL-Schnittstelle aufzurufen. Es handelt sich dabei um folgende Programmiersprachen:

- COBOL (ab Version 2.0)
- FORTRAN (ab Version 2.2)
- PL/1 (ab Version 4.1)

- C (ab Version 2.0)

Der Aufruf von DIV-Funktionen über eine CALL-Schnittstelle setzt die sog. ILCS-Linkage voraus.
Der Adapter Window Services wird ausgeliefert als Laufzeit-Bibliothek (OML) in einer Datei mit dem Namen SYSLIB.DWS.120 unter der Kennung TSOS der Systembetreuung.
Bei Aufruf von DIV über Window Services können einige DIV-Funktionen nicht oder nur eingeschränkt genützt werden.

Für nähere Information über die CALL-Schnittstelle siehe Anhang im Handbuch „DVS-Makros“ [1] .

18.1 DIV-Arbeitsweise

Die Aktionen von DIV werden intern über einen Seitenzustandsanzeiger gesteuert. Folgende Seitenzustände sind zu unterscheiden: FRESHLY_OBTAINED (F), ACCESSED (A) und MODIFIED (M).

Was bei einem Zugriff auf eine Seite in einem DIV-Fenster geschieht, hängt davon ab, in welchem Zustand sich die Seite befindet.

Steuerung durch Seitenzustände

Nach der Definition eines Fensters mit der Maßgabe, dass im Fenster die Seiten der Datei erscheinen sollen (Funktion MAP mit Operand DISPOS=*OBJECT), befinden sich alle Seiten des Fensters im Zustand FRESHLY_OBTAINED. Das ist der Zustand, in dem sich eine Seite auch nach Speicherplatzallokierung mit dem Makro REQM befindet (Initialzustand).

Bei einem Zugriff auf eine solche Seite (durch einen beliebigen CPU-Befehl) gibt es einen Page Fault Interrupt und DIV erhält von der Speicherverwaltung die Kontrolle, um die Seite aus der Datei einzulesen. Liegt die Seite hinter der logisch letzten Dateiseite, wird sie mit X'00' initialisiert. Danach befindet sich die Seite nicht mehr im Initialzustand, sondern im Zustand ACCESSED. Bei weiteren Zugriffen wird sie wie eine beliebige andere Seite im virtuellen Adressraum behandelt.

Wird die Seite vom Programm verändert, befindet sie sich im Zustand MODIFIED. Eine so gekennzeichnete Seite hat einen von der entsprechenden Datei-Seite verschiedenen Inhalt. Durch die DIV-Funktion SAVE werden modifizierte Seiten in die Datei geschrieben. Danach besitzen diese Seiten nicht mehr den Seitenzustand MODIFIED und werden von einem folgenden SAVE nicht in die Datei geschrieben, es sei denn, sie wurden durch einen Zugriff des Programms erneut geändert.

Welche Fensterseiten in die Datei auf Platte zurückgeschrieben werden, hängt darüber hinaus vom Wert des Operanden DISPOS beim Anlegen des Fensters (Funktion MAP) und der Lage der Seite bezüglich des logischen Dateiendes ab.

Von der DIV-Funktion RESET wird eine modifizierte Seite in den Initialzustand gebracht, was bei einem mit DISPOS=*OBJECT spezifizierten Fenster bewirkt, dass die Seite bei einem folgenden Zugriff erneut aus der Datei eingelesen wird (bzw. mit X'00' initialisiert wird, wenn sie jenseits des logischen Dateiendes liegt). RESET hat daher bei einem mit DISPOS=*OBJECT definierten Fenster die Wirkung, dass erneut die Seiten der Datei im Fenster erscheinen.

Die Funktion RESET mit dem Operanden RELEASE=*YES bewirkt, dass nicht nur modifizierte Seiten, sondern alle Seiten eines definierten Bereichs in den Initialzustand gebracht werden. Die Folge ist, dass alle diese Seiten bei einem Zugriff aus der Datei gelesen werden.

Bei einem mit DISPOS=*UNCHNG definierten Fenster ändert MAP die Seitenzustände nicht. Wurde auf eine Seite noch nie zugegriffen, bleibt sie im Initialzustand, andernfalls befindet sie sich (aus der Sicht von DIV) im Zustand MODIFIED.

Wenn durch einen Zugriff auf eine Seite eines mit DISPOS=*UNCHNG spezifizierten Fensters DIV die Kontrolle erhält, wird die Seite mit X'00' initialisiert und befindet sich danach im Zustand MODIFIED.

Falls eine Seite mit SAVE schon einmal in die Datei geschrieben und danach durch RESET in den Initialzustand gebracht wurde, wird sie auch bei einem mit DISPOS=*UNCHNG spezifizierten Fenster aus der Datei gelesen und nicht mit X'00' initialisiert.

Hinweis

Adressraum allokieren: Um mit DIV arbeiten zu können, muss der Benutzer virtuellen Adressraum anfordern. Adressraum im Programmraum wird mit dem Makro REQM allokiert; Adressraum im Datenraum mit dem Makro DSPSRV. Bei Zuteilung von Hauptspeicherseiten werden diese vorher mit X'00' initialisert.

18.2 Dateiverlängerung und Dateiverkürzung

Das logische Dateiende kann von der SAVE-Funktion verändert werden.

18.2.1 Datei logisch verlängern

Regeln zur logischen Dateiverlängerung:

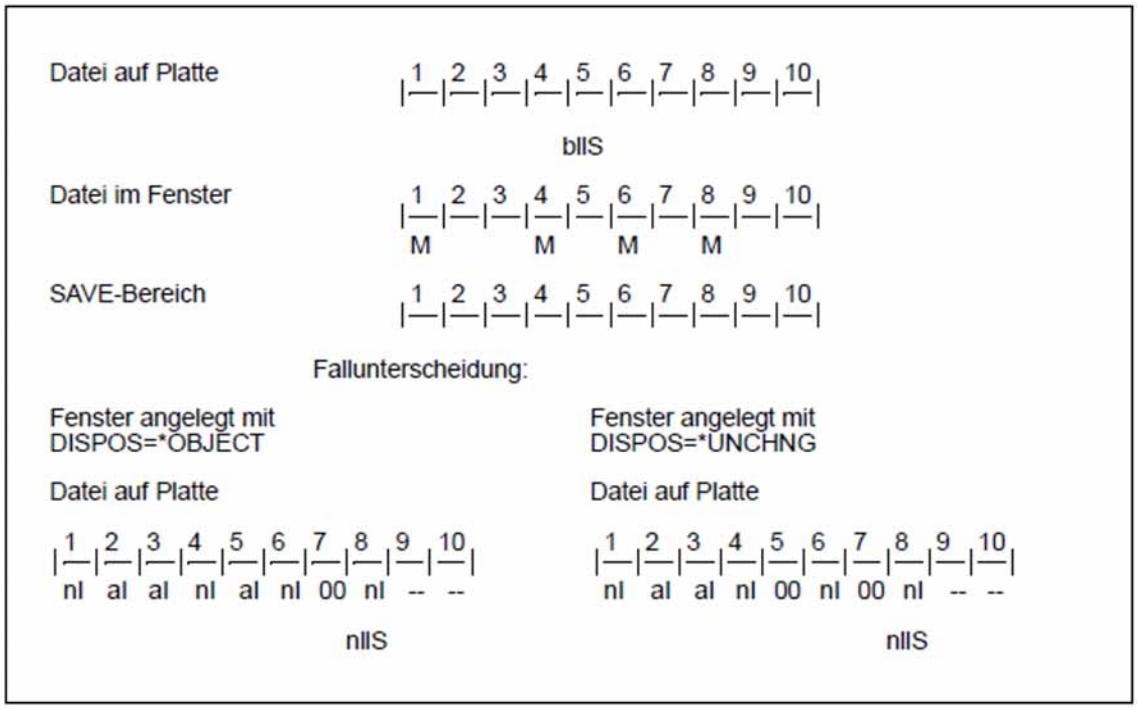
- Eine Datei wird bis zur letzten Seite im Zustand MODIFIED verlängert, die hinter der bisher logisch letzten Seite liegt und im SAVE-Bereich und in einem Fenster enthalten ist.
- Seiten, die im neuen Dateibereich liegen, um den die Datei verlängert wird, werden in die Datei geschrieben, wenn sie sowohl im SAVE-Bereich als auch in einem Fenster enthalten sind.

  Nicht-modifizierte Seiten im neuen Dateibereich werden mit X'00' geschrieben.

  Seiten, die nicht gleichermaßen im SAVE-Bereich und in einem Fenster enthalten sind, werden nicht in die Datei geschrieben.
- Das DISPOS-Attribut des Fensters (bzw. der Fenster) spielt bei einer Dateiverlängerung keine Rolle.

Die logisch letzte Dateiseite ist die logisch letzte Dateiseite aus der Sicht der eigenen Task. Eine andere Task kann (bei SHARUPD=*YES) die logisch letzte Seite verändert haben, ohne dass die Veränderung der eigenen Task bekannt wurde.
Auf diese Weise kann eine Dateiverlängerung einer parallelen Task wieder rückgängig gemacht werden.

Beispiel 1: Datei logisch verlängern



Legende

1...10	Seiten-Nummer (Datei/Fenster)
al	alter Inhalt
bllS	bisher logisch letzte Seite
M	modifiziert

nI	neuer Inhalt
nllS	neue logisch letzte Seite
--	undefinierter Inhalt

Erläuterung

Die Datei ist physikalisch 10 Seiten (1 Seite = 4 KB) lang. Das logische Dateiende ist die Seite 5. Der SAVE-Bereich umfasst die Seiten 1-10. Die letzte modifizierte Fensterseite des SAVE-Bereiches ist die Seite 8.

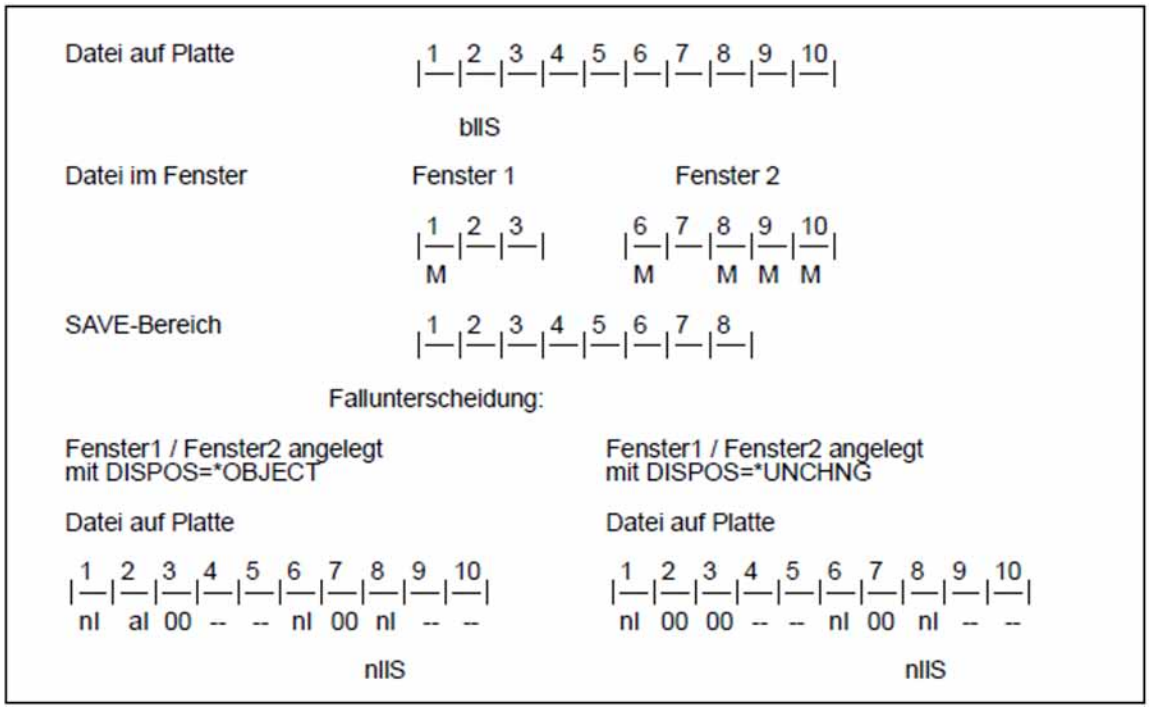
Unabhängig von der DISPOS-Eigenschaft des Fensters werden zwischen der bisher logisch letzten Seite und der neuen logisch letzten Seite alle modifizierten Seiten in die Datei auf Platte geschrieben und alle nicht-modifizierten Seiten mit X'00' in der Plattendatei initialisiert.

Für den übrigen Bereich (Seite 1 bis bisher logisch letzte Seite) ist zu unterscheiden:

- DISPOS=*OBJECT:
  Es werden nur die modifizierten Seiten in die Datei geschrieben.
- DISPOS=*UNCHNG:
  Modifizierte Seiten werden ebenfalls in Datei auf Platte geschrieben.

Nicht-modifizierte Seiten, die noch nicht (durch SAVE) in die Datei geschrieben wurden, werden in der Datei mit X'00' initialisiert (Seite 5). Sie erhalten danach den Seitenzustand SAVED.
Im Beispiel wird angenommen, dass die Seiten 2 und 3 schon einmal durch SAVE in die Datei geschrieben wurden. Für nicht-modifizierte Seiten, die schon einmal durch SAVE in die Datei geschrieben wurden (Seite 2 und Seite 3), werden keine Aktionen durchgeführt.

Beispiel 2: Datei logisch verlängern



Legende

1...10	Seiten-Nummer (Datei/Fenster)
aI	alter Inhalt
bllS	bisher logisch letzte Seite
M	modifiziert
nI	neuer Inhalt
nllS	neue logisch letzte Seite
--	undefinierter Inhalt

Erläuterung

Die Datei ist 10 Seiten lang. Das logische Dateiende ist die Seite 2. Die Dateiseiten 1-3 werden dem Fenster 1 zugeordnet, die Seiten 6-10 dem Fenster 2.
Es wird ein 8 Seiten langer SAVE-Bereich definiert. Er erstreckt sich über Fenster 1 und die ersten drei Seiten von Fenster 2.

Die letzte modifizierte Fensterseite des SAVE-Bereichs ist die Seite 8. Seite 8 liegt hinter dem bisher logischen Dateiende und wird daher zum neuen logischen Dateiende (neue letzte logische Seite).

Unabhängig von der DISPOS-Eigenschaft des Fensters werden zwischen der bisherigen logisch letzten Seite und der neuen logisch letzten Seite alle modifizierten Seiten in die Datei auf Platte geschrieben (Seite 6 und 8) und alle nicht-modifizierten Fensterseiten mit X'00' in der Plattendatei initialisiert (Seite 3 und 7).

Für den übrigen Bereich (Seite 1 bis bisherige logisch letzte Seite) ist zu unterscheiden:

- DISPOS=*OBJECT:
  Es werden nur die modifizierten Seiten in die Datei geschrieben (Seite 1).
- DISPOS=*UNCHNG:
  Modifizierte Seiten werden ebenfalls in Datei auf Platte geschrieben.
  Nicht-modifizierte Seiten, die noch nicht (durch SAVE) in die Datei geschrieben wurden, werden in der Datei mit X'00' initialisiert (Seite 2). Sie erhalten danach den Seitenzustand SAVED.
  (Für nicht-modifizierte Seiten, die schon einmal durch SAVE in die Datei geschrieben wurden, werden keine Aktionen durchgeführt, siehe auch vorhergehendes Beispiel).

Die Seiten 4 und 5 bleiben in der Datei unverändert, da für sie kein Fenster existiert. Da sie hinter dem bisherigen logischen Dateiende liegen, ist ihr Inhalt undefiniert.

Die Änderungen auf Seite 9 und 10 in Fenster 2 werden nicht auf die Platte zurückgeschrieben, da diese Seiten nicht im SAVE-Bereich liegen.

18.2.2 Datei logisch verkürzen / Dateiinhalt löschen

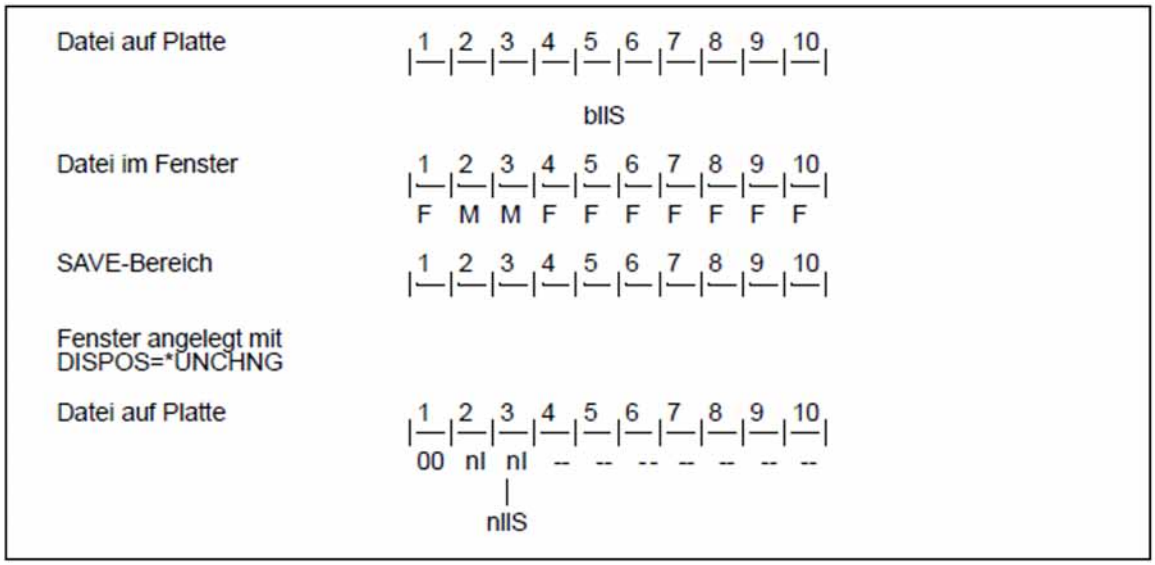
Eine Datei wird logisch gekürzt, wenn alle der folgenden Bedingungen erfüllt sind:

1. Es liegt keine Bedingung zur Dateiverlängerung vor.
2. Die logisch letzte Seite befindet sich in einem Fenster mit dem Attribut DIS-POS=*UNCHNG und im SAVE-Bereich.
3. Zumindest die letzte Seite befindet sich im Initialzustand und ist noch nicht durch SAVE in die Datei geschrieben worden.

Eine Dateiverkürzung endet, wenn eines der folgenden Ergebnisse vorliegt:

- das Ende des SAVE-Bereichs wurde erreicht
- mindestend eine Seite liegt nicht in einem Fenster mit dem Attribut DIS-POS=*UNCHNG (kein Fenster oder Fenster mit DISPOS=*OBJECT)
- mindestens eine Seite ist nicht im Initialzustand oder wurde schon einmal mit SAVE in die Datei geschrieben

Beispiel 1: Datei logisch verkürzen



Legende

1...10	Seiten-Nummer (Datei/Fenster)
aI	alter Inhalt
bllS	bisher logisch letzte Seite
F	FRESHLY_OBTAINED
M	MODIFIED
nI	neuer Inhalt
nllS	neue logisch letzte Seite

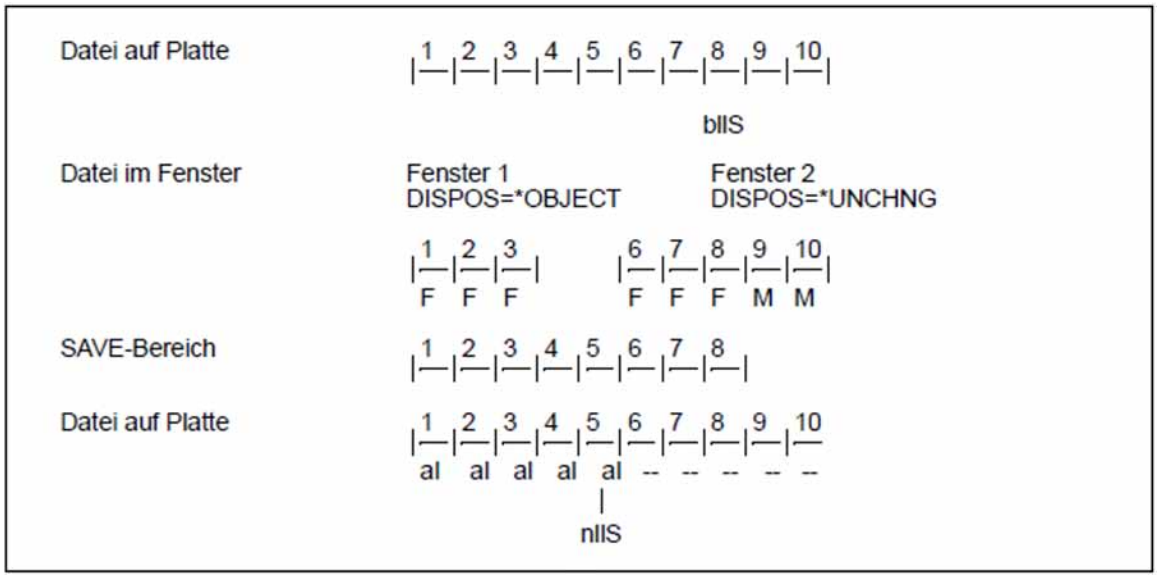
--	undefinierter Inhalt

Erläuterung

Auf die Fensterseiten 4-10 wurde noch nie zugegriffen oder sie wurden durch RESET in den Initialzustand (FRESHLY_OBTAINED) gebracht.
Seite 5 bildet das logische Dateiende. Auf die unmittelbar davor liegende Seite wurde noch nie zugegriffen (Seite 4). Die Datei wird durch SAVE um die Seiten 4 und 5 logisch verkürzt.

Vom Dateianfang bis zum neuen logischen Dateiende (Seiten 1 bis 3) werden modifizierte Seiten in die Datei geschrieben. Seiten in diesem Bereich, auf die noch nie zugegriffen wurde (Seitenzustand FRESHLY_OBTAINED; noch nicht durch eine vorausgegange SAVE-Operation auf Platte geschrieben) werden in der Plattendatei gelöscht, d.h. mit X'00' initialisiert. Im Beispiel ist dies die Seite 1.

Beispiel 2: Datei logisch verkürzen



Legende

1...10	Seiten (Datei/Fenster)
aI	alter Inhalt
bllS	bisher logisch letzte Seite
M	modifiziert
nI	neuer Inhalt
nllS	neue logisch letzte Seite
--	undefinierter Inhalt

Erläuterung

Jenseits der logisch letzten Seite (Seite 8) sind keine modifizierten Seiten im SAVE-Bereich. Daher ist keine Bedingung für eine Dateiverlängerung gegeben.
Da auf die logisch letzte Seite, d.h. die letzte Seite des SAVE-Bereichs (Seite 8) und die vorausgehenden Seiten des zweiten Fensters noch nicht zugegriffen wurde, sind diese Seiten im Initialzustand. Die Datei wird um diese Seiten verkürzt.
Die Verkürzung endet, wenn eine Seite entweder in keinem Fenster erscheint oder nicht in einem Fenster mit DISPOS=*UNCHNG.
Im Beispiel wurden die Seiten 1, 2 und 3 schon einmal durch SAVE in die Datei geschrieben. Daher behalten sie ihren alten Inhalt und werden nicht mit X'00' intitialisiert.

Für den Fall, dass Seite 9 zum SAVE-Bereich gehört, wird die Datei bis zu einschließlich dieser Seite verlängert.

18.2.3 Datei physikalisch verlängern oder verkürzen

Das physikalische Dateiende kann von der MAP-Funktion verändert werden.

Datei physikalisch verlängern

Bei der Definition eines Fensters durch MAP sorgt DIV (bei OPEN OUTIN/INOUT, nicht IN-PUT) dafür, dass für den Dateibereich, der vom Fenster repräsentiert wird, auf der Platte Speicherplatz (4-KB-Blöcke) allokiert ist. Für Fensterseiten, die hinter dem physikalischen Dateiende liegen, wird von DIV Speicherplatz (4-KB-Blöcke) allokiert.

Datei physikalisch verkürzen

Durch den Makro FILE mit einer negativen SPACE-Angabe oder mit dem Kommando MODIFY-FILE-ATTRIBUTES ... RELEASE= können von einer Datei belegte Blöcke freigegeben werden. Eine Freigabe endet spätestens bei der logisch letzen 4-KB-Seite.

18.3 Multi-User-Betrieb

Eine PAM-Datei kann mit den folgenden Zugriffsmethoden erstellt bzw. bearbeitet werden:

- DIV
- UPAM (siehe Kapitel „UPAM – User Primary Access Method“ (UPAM - User Primary Access Method))
- FASTPAM (siehe Kapitel „FASTPAM – Fast Primary Access Method“ (FASTPAM - Fast Primary Access Method))

FASTPAM und DIV können jedoch nur UPAM-Dateien mit der Eigenschaft BLKCTRL=NO bearbeiten.
Die Erlaubnis für eine parallele Dateibearbeitung ist von den bei der Eröffnung angegebenen Operandenwerten für SHARUPD, MODE, LOCKENV und LOCVIEW abhängig.

Möglichkeiten der parallelen Eröffnung

			USER B									
	SHARUPD=		***YES**			***NO**			***WEAK**			
		OPEN-Modus	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	LMAP	INOUT	OUTIN
USER A	***YES**	INPUT INOUT OUTIN	X O O	O O O		X			X X X	X		
	***NO**	INPUT INOUT OUTIN	X			X			X X X	X		
	***WEAK**	INPUT LMAP INOUT OUTIN	X X	X		X X	X		X X X X	X X O	X O	

Tabelle 44: DIV: erlaubte SHARUPD-/OPEN-Kombinationen

LMAP: INPUT LOCVIEW=MAP (gibt es nur bei DIV)

X: OPEN erlaubt

O: OPEN nur erlaubt,

- wenn die Eröffner dieselbe blockorientierte Zugriffsmethode benutzen (nur UPAM/FASTPAM oder nur DIV)
- **und** denselben Wert für den Operanden LOCKENV benutzen (alle LOCKENV=*HOST oder LOCKENV=*XCS)
- **und** alle im selben HOST laufen *oder* im selben XCS-Verbund bei Verwendung von LOCKENV=*XCS

Hinweise

- Leseoperationen mit SHARUPD=*WEAK können eine Datei gleichzeitig mit jeder beliebigen Schreiboperation eröffnet haben.

  Den Leseoperationen mit DIV-SHARUPD=*WEAK, die bei OPEN LOCVIEW=*MAP spezifiziert haben, werden bereits bei MAP alle Fensterseiten aus der Datei in das Fenster eingelesen, wobei von DIV sichergestellt wird, dass während des Einlesens keine Dateiseiten durch einen parallelen SAVE einer Schreiberoperation mit DIV-SHARUPD=*WEAK verändert werden können.

  Bei einer UPAM/FASTPAM-Schreiboperation existiert dieser Schutz gegen paralleles Schreiben nicht.Aus diesem Grund sind Leseoperationen mit DIV-SHARUPD=*WEAK, für die LOCVIEW=*MAP spezifiziert wurde, nur mit Schreiboperationen mit DIV-SHARUPD=*WEAK verträglich, nicht jedoch mit anderen Schreiboperationen.

  Auch die übrigen oben für den Eintrag 'O' formulierten Bedingungen müssen erfüllt sein (alle Eröffner mit gleichen Werten für den LOCKENV-Operanden und alle Eröffner im gleichen Host oder – wenn in verschiedenen Hosts – mit der Angabe LOCKENV=*XCS).

  Leseoperationen mit DIV-SHARUPD=*WEAK, die bei OPEN LOCVIEW=*NONE spezifiziert haben, besitzen dieselbe Verträglichkeit wie Leseoperationen mit UPAM/FASTPAM-SHARUPD=*WEAK.
- Eröffner mit DIV-SHARUPD=*YES sind nicht mit Eröffnern mit UPAM/FASTPAM-SHARUPD=*YES verträglich.
- Leseoperationen sind immer miteinander verträglich (unabhängig von Zugriffsmethode, SHARUPD-Spezifikation, LOCKENV-Spezifikation und Host).
- Bei Zugriffen auf Dateien im Modus SHARUPD=YES kann der Fall eintreten, dass eine Datei mit einer Dateigröße < 32 GB durch entsprechende Verarbeitung zu einer Datei >= 32 GB wird.

  Hier werden zwei Fälle unterschieden:

  - Aufrufer, die auf diese Situation vorbereitet sind (mit der Angabe LARGE_FILE=*ALLOWED beim Makro FCB bzw. EXCEED-32GB=*ALLOWED beim Kommando ADD-FILE-LINK)
  - nicht vorbereitete Aufrufer (Angabe LARGE_FILE=*FORBIDDEN beim Makro FCB bzw. EXCEED-32GB=*FORBIDDEN beim Kommando ADD-FILE-LINK).

  Es wird nach jedem Aufruf des Allocators die Größe der betroffenen Datei überprüft. Wenn bei dieser Überprüfung eine Dateigröße >= 32 GB ermittelt wird und im zugehörigen FCB das Attribut LARGE_FILE=*FORBIDDEN gesetzt ist, wird die Verarbeitung abgebrochen und ein entsprechender Returncode in der eigenen Parameterliste DIV(I) bzw. eine DMS-Meldung ausgegeben.

18.4 Datenkonsistenz und Datenaktualität

Konsistenz der Daten in der Plattendatei beim Multi-User-Betrieb

Durch die Angabe SHARUPD=*YES bei der Funktion OPEN erreicht der Benutzer, dass mehrere Schreiber eine Datei gleichzeitig bearbeiten können.

> ¡ Schreiboperationen (SAVE) verschiedener Benutzer werden **nicht** von DIV koordiniert, d.h. der Benutzer ist für die Synchronisierung zuständig.

Konsistenz der Daten in der Plattendatei nach einem Systemausfall

Wenn SAVE durch einen Systemausfall (oder durch einen Taskabsturz) abgebrochen wird, kann es sein, dass einige modifizierte Seiten bereits in die Datei geschrieben wurden, andere noch nicht.

Der Inhalt der Plattendatei kann also in einem solchen Fall in einem inkonsistenten Zustand sein.

Konsistenz der Daten in einem Fenster beim Multi-User-Betrieb

In einem Fenster können inkonsistente Daten durch parallele Tasks nur dann erscheinen, wenn ein paralleler Schreiber die Seiten des entsprechenden Dateibereichs verändert und eine geänderte Seite nach einem Zugriff im Fenster erscheint.

Folgende Eröffner sind von Änderungen von Fensterseiten durch einen parallelen Schreiber nicht betroffen:

- SHARUPD=*NO-Eröffner (es kann keinen parallelen Schreiber geben),
- SHARUPD=*WEAK-Schreiber (es kann keinen parallelen Schreiber geben),
- SHARUPD=*WEAK-Leser, falls bei OPEN der Operand LOCVIEW=*MAP spezifiziert ist.

DIV stellt dann sicher, dass während des Einlesens der Fensterseiten zum MAP-Zeitpunkt keine Seiten von einem parallelen Schreiber in der Datei (durch SAVE) verändert werden können. Nachdem die Seiten eingelesen sind, können sie von einem parallelen Schreiber verändert werden.

> ¡ Wenn ein paralleler Schreiber das logische Dateiende verändert, kann nicht davon ausgegangen werden, dass bei anderen Eröffnern die Änderung berücksichtigt wird (z.B. bei einem Fensterzugriff oder bei SAVE).

Konsistenz der Daten in einem Fenster nach einem Systemausfall

Modifizierte Daten, die nur im Fenster modifiziert und noch nicht in die Plattendatei zurückgeschrieben wurden, sind verloren.

Aktualität der Daten in einem Fenster

In einem Fenster können Daten nur dann an Aktualität verlieren, wenn ein paralleler Schreiber die Seiten des entsprechenden Dateibereichs verändert und eine geänderte Seite nach einem Zugriff im Fenster erscheint.

In diesem Zusammenhang ist die Beachtung folgender Eigenschaften von DIV wichtig:

- Eine Fensterseite wird beim ersten Zugriff aus der Datei eingelesen und ist im weiteren Ablauf nur evtl. Änderungen durch CPU-Befehle unterworfen.
  Erst nach einem RESET wird sie bei einem erneuten Zugriff aus der Datei eingelesen.

- Ob eine Seite bei einem Zugriff einzulesen ist, hängt auch davon ab, ob es überhaupt eine entsprechende Dateiseite gibt, d.h. ob die Seite vor oder nach dem logischen Dateiende liegt.

  > **i** Wann die Veränderung des logischen Dateiendes durch einen parallelen Schreiber für einen anderen Eröffner wirksam wird, ist implementierungsabhängig. Der Benutzer darf sich nicht darauf verlassen, dass Änderungen des logischen Dateiendes durch einen parallelen Schreiber auch in seinen Fenstern sichtbar werden.

19 EAM - Evanescent Access Method

EAM ist die Zugriffsmethode für EAM-Dateien im BS2000 und hat folgende Eigenschaften:

- EAM-Dateien werden nicht katalogisiert. Das Eröffnen einer EAM-Datei erfordert deshalb keinen Plattenzugriff.
- Jede EAM-Datei wird nach Beendigung des sie eröffnenden Auftrags automatisch gelöscht (temporäre Datei).
- Die Verständigung zwischen EAM und dem Benutzer erfolgt nur über den EAM-Steuerblock (MFCB = Mini File Control Block). Modifizierung des MFCB zum Eröffnungszeitpunkt ist nicht vorgesehen.
- Es findet keine Kennsatzverarbeitung statt.
- EAM benutzt nur gemeinschaftliche Datenträger (Pubsets). Dabei spielt es keine Rolle, ob es sich um Platten mit oder ohne PAM-Schlüssel (K- oder NK-Platten) handelt.
- Der Speicherplatzbedarf für die EAM-Routinen und die Laufzeiten für die Lese/Schreibzugriffe sind geringer als bei den Zugriffsmethoden für katalogisierte Dateien.
- Eine EAM-Datei kann nur von dem sie aufrufenden Auftrag bearbeitet werden, aber ein Auftrag kann mehrere EAM-Dateien eröffnen und bearbeiten.
- EAM ist eine blockorientierte Zugriffsmethode, Grundlage der Verarbeitung ist ein 2048 Byte langer Block. Eine Satzstruktur bleibt unbekannt. Bei geketteter Ein-/Ausgabe können bis zu 16 aufeinander folgende Blöcke mit einem Makroaufruf bearbeitet werden.
- Bei Programm-Wiederanlauf mit RESTART-PROGRAM werden alle EAM-Dateien des Auftrags gelöscht.

EAM-Funktionen

Die EAM-Funktionen werden durch den Aufruf des Makros EAM genutzt. Es gibt folgende Grundfunktionen:

Funktion	Bedeutung
CHECK CHECK_WAIT	Prüfen und Warten auf Beendigung der Ein-/Ausgabe
CLOSE CLOSE_ALL	Schließen einer Datei
ERASE ERASE_ALL	Löschen einer Datei
OPEN	Einrichten und Eröffnen einer neuen Datei
READ	Lesen (blockweise, sequenziell oder direkt)
REOPEN	Eröffnen einer bestehenden Datei
WRITE	Schreiben (blockweise, sequenziell oder direkt

Tabelle 45: EAM-Makro (Funktionen)

Eine Operation wird durch die Angabe eines sedezimalen Operationsschlüssels im MFCB ausgewählt und mit dem EAM-Makroaufruf angestoßen. Die Wirkung ist durch die MFCB-Felder bestimmt, die EAM nach Analyse des Operationsschlüssels zusätzlich auswertet.

19.1 EAM-Verarbeitung

- Verwendung von Prüfoperationen
- Änderung von Verarbeitungseigenschaften
- Überlappte Ein-/Ausgabe
- Behandlung von Bindemoduldateien durch EAM

19.1.1 Verwendung von Prüfoperationen

Nach einem Lese- oder Schreibaufruf erhält der Benutzer die Steuerung zurück, sobald die angeforderte Operation akzeptiert wird. Diese Operation braucht also noch nicht beendet zu sein.

Bevor eine Lese- oder Schreiboperation angestoßen wird, wird die Beendigung einer eventuell vorhergehenden Lese- oder Schreiboperation abgewartet (mit einer impliziten Prüf- und Warteoperation). Ebenso wird die Beendigung der letzten Lese-/Schreiboperation abgewartet, wenn eine Schließoperation angefordert wurde.

Eine Prüfoperation ist also nur notwendig nach der letzten Operation in einer Reihe von Lese-/Schreiboperationen, wenn die Datei nicht sofort wieder geschlossen wird oder wenn mit geketteter Ein-/Ausgabe bis zum Erreichen der Dateiendebedingung gelesen wird (im Fehlerbyte: IDMFIB oder IDMFE =1) und die Anzahl der übertragenen Blöcke ungleich 0 ist.

Beispiel

In einer EAM-Datei werden 3 Leseoperationen durchgeführt. Die Datei wird danach nicht wieder geschlossen, weil sie für spätere Ein-/Ausgaben noch benötigt wird. Diese Ein-/Ausgabeoperationen werden jedoch erst nach der Verarbeitung der gelesenen Blöcke angefordert:

```
LESEN
:
PRÜFEN/WARTEN                       Die Beendigung der letzten Ein-/Ausgabe-
                                    Operation wird abgewartet.
Verarbeitung der gelesenen Blöcke
:
weitere Ein-/Ausgabeoperationen
```

19.1.2 Änderung von Verarbeitungseigenschaften

Beim Eröffnen oder Wiedereröffnen einer Datei werden im Systemspeicher folgende Angaben sichergestellt:

- gekettete/nichtgekettete Ein-/Ausgabe
- Anzahl der zu übertragenden Blöcke

Soll eine dieser Angaben während der Dateiverarbeitung geändert werden, ist folgendes Vorgehen erforderlich:

1. Datei schließen
2. Felder im MFCB modifizieren
3. Datei wieder eröffnen

Die Anzahl der zu übertragenden Blöcke ist z.B. dann zu ändern, wenn mit der letzten Schreiboperation weniger Blöcke übertragen werden sollen, als beim Eröffnen der Datei angegeben wurde.

Beispiel

In eine EAM-Datei sind 99 Blöcke zu schreiben. Es sollen jeweils 15 Blöcke gekettet übertragen werden (Byte IDMFNHP im MFCB). Folgende Operationen werden dann angefordert:

```
ÖFFNEN (neue Datei) -> SCHREIBEN -> SCHREIBEN -> SCHREIBEN -> SCHREIBEN ->
SCHREIBEN -> SCHREIBEN -> SCHLIESSEN *) -> WIEDERERÖFFNEN -> SCHREIBEN ->
WIEDERERÖFFNEN -> SCHREIBEN -> SCHLIESSEN
```

*) Nach Ausführung der sechs Schreiboperationen sind noch 9 Blöcke zu schreiben. Deshalb muss die Datei geschlossen und wieder eröffnet werden mit dem Wert 9 für die Anzahl zu übertragender Blöcke.

19.1.3 Überlappte Ein-/Ausgabe

Die Verarbeitungszeit kann durch asynchrone Ein-/Ausgaben verkürzt werden. Nach Anstoßen einer Ein-/Ausgabeoperation wird die Steuerung sofort an das Programm zurückgegeben, wo parallel zur physikalischen Ein-/Ausgabe weitere Verarbeitung stattfindet.
Die nächste Ein-/Ausgabeoperation wird angestoßen für einen zweiten, sich mit dem ersten Bereich nicht überschneidenden Ein-/Ausgabebereich ... usw.

Bild 24 zeigt die Überlappung von Verarbeitung und Eingabeoperationen.

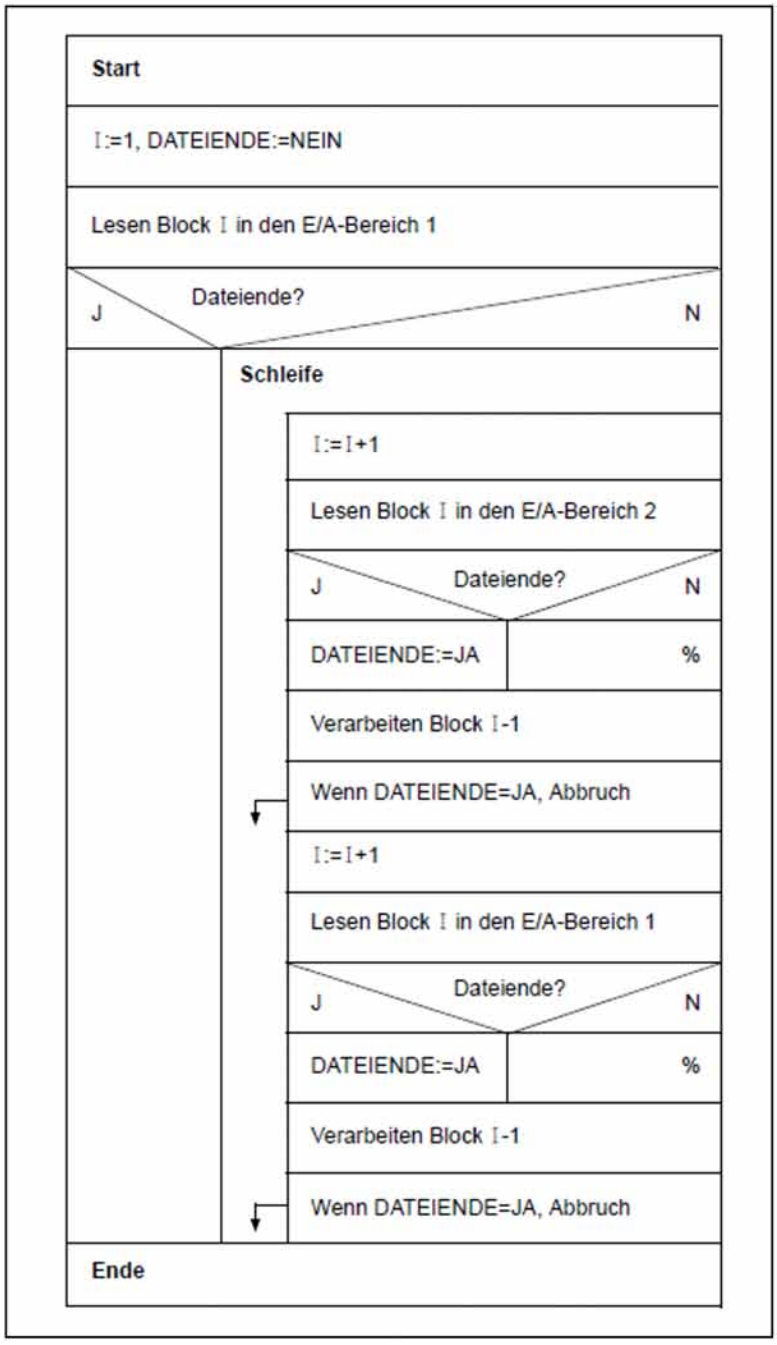


Bild 24: Überlappte Ein-/Ausgabe bei der Zugriffsmethode EAM

19.1.4 Behandlung von Bindemoduldateien durch EAM

Jeder Auftrag kann genau eine Bindemoduldatei bearbeiten. Ist das Bit IDMFOO des Zusatzbytes gesetzt, beziehen sich alle Operationen auf die Bindemoduldatei.
Die Aktionen beim Eröffnen oder Wiedereröffnen einer Datei sind dem folgenden Diagramm zu entnehmen:

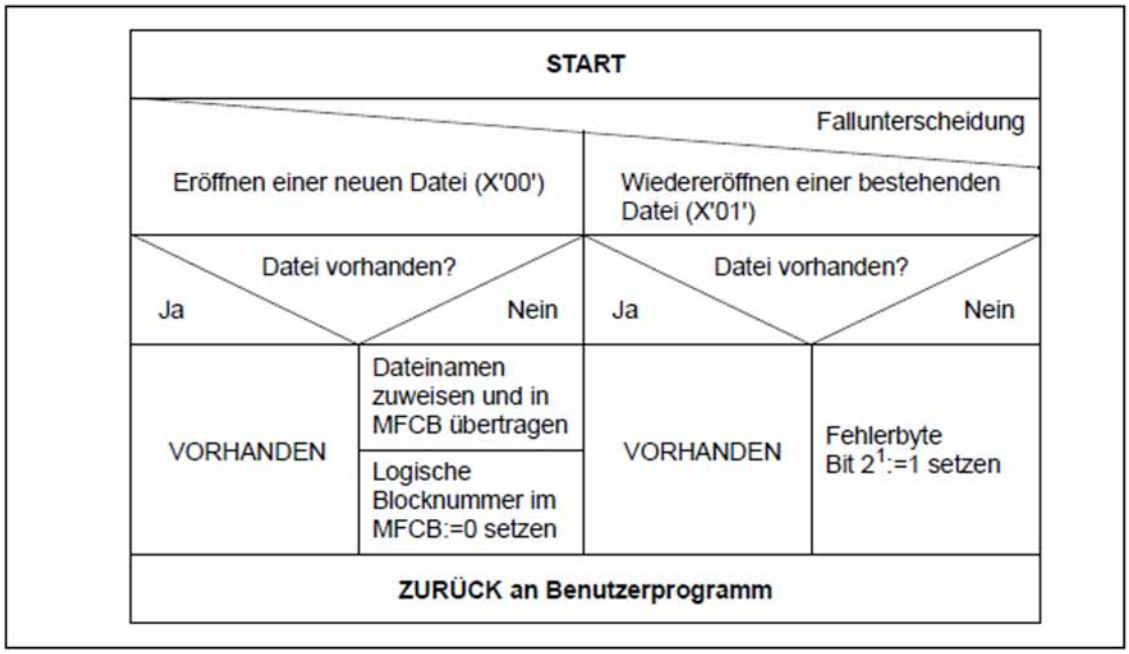


Bild 25: Aktionen bei EAM-Dateieröffnung

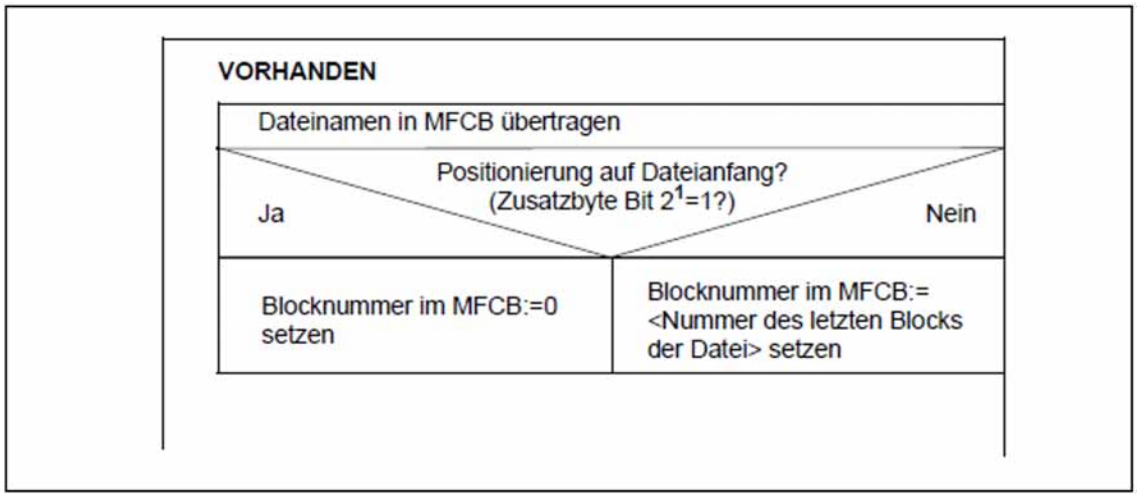


Bild 26: Ablauf beim Eröffnen der Bindemoduldatei

20 FASTPAM - Fast Primary Access Method

FASTPAM ist eine Blockzugriffsmethode für NK4-Plattendateien. Sie bietet eine mit UPAM vergleichbare Funktionalität, wobei jedoch die Performance gegenüber UPAM erheblich verbessert wurde und Multi-Server-Systeme besonders berücksichtigt werden. FASTPAM zeichnet sich aus durch:

- performante Ein-/Ausgaben auf Grund von Einsparungen von CPU-Befehlen
- eine schmale, übersichtliche Schnittstelle
- Unterstützung von Ein-/Ausgaben in Data Spaces

FASTPAM bietet eine Untermenge der Funktionalität von UPAM.

Die Performance-Gewinne resultieren im Wesentlichen daraus, dass Aktionen zur Vorbereitung von Ein-/Ausgaben aus dem Ein-/Ausgabe-Pfad herausgezogen und vor dem OPEN abgewickelt und unabhängig von einer einzelnen Ein-/Ausgabe durchgeführt werden.
Hierzu gehören:

- Parameterlisten resident anlegen
- Ein-/Ausgabepfade vorbereiten
- Pufferbereiche resident anlegen

Die Ein-/Ausgabepfade bestehen aus dem gesamten vom System für eine Ein-/Ausgabe benötigten, residenten Speicher, zusammen mit allen beim Einrichten des Environments vorgenerierbaren Daten. In diesem Speicher werden z.B. vor den Ein-/Ausgaben die Kanalprogramme aufgebaut. Benutzerseitig wird der Parameterlisten- und IO-Area-Pool-Speicher fixiert (d.h. er ist nicht mehr seitenwechselbar und kann vom Benutzer nicht mehr freigegeben werden).

Über FASTPAM-Verwaltungsaufrufe richtet der Benutzer zwei Bereiche ein:

- FASTPAM-Environment
- FASTPAM-IO-Area-Pool

Das FASTPAM-Environment dient zur Aufnahme der FASTPAM-Parameterlisten, mit denen Ein-/Ausgabe-Aufträge erteilt werden. Außerdem gehören zum FASTPAM-Environment auch noch Speicherbereiche des Betriebssystems (Ein-/Ausgabepfade).

Der FASTPAM-IO-Area-Pool ist ein Bereich mit 4-KB-Puffern.

Falls der Aufrufer über die FASTPAM-Berechtigung verfügt (Eintrag im Benutzerkatalog), können diese Bereiche speicherresident gehalten werden. Diese FASTPAM-Bereiche können dateiübergreifend und, falls sie in Common-Memory-Pools liegen, auch task-übergreifend verwendet werden, sodass eine effiziente Nutzung des Betriebsmittels „residenter Speicher“ möglich ist. Die Größe des FASTPAM-Environments wird durch die Maximalzahl parallel durchzuführender Ein-/Ausgaben bestimmt.

Bei der FASTPAM-Schittstelle handelt es sich um eine SVC-Schnittstelle. Die Aufträge werden durch eine Parameterliste formuliert; Rückmeldungen über das Ergebnis erfolgen über einen Returncode in der Parameterliste (nicht über Exits).

Mit FASTPAM ist es möglich, Ein-/Ausgaben direkt in Datenräumen (Data Spaces) durchzuführen. Dazu werden IO-Area-Pools in Datenräume gelegt; allerdings lassen sich diese IO-Area-Pools nur nicht-resident anlegen.

Die Schnittstelle von FASTPAM bietet zwei Gruppen von Funktionen:

- Verwaltungsfunktionen (Makro FPAMSRV)
  - Anlegen eines neuen FASTPAM-Environments bzw. Anschließen an bestehende FASTPAM-Environments und FASTPAM-IO-Area-Pools
  - Abbau bzw. Trennen einer Task von FASTPAM-Environments und FASTPAM-IO-Area-Pools
  - Öffnen und Schließen von Dateien
- Zugriffsfunktionen (Makro FPAMACC)
  - synchrones Lesen und Schreiben von logischen Blöcken
  - asynchrones Lesen und Schreiben von logischen Blöcken
  - Warten auf die Beendigung asynchroner Ein-/Ausgabe-Aufträge
  - Benachrichtigung über die Beendigung asynchroner Ein-/Ausgabe-Aufträge

Als Anwendungsbereiche kommen in erster Linie systemnahe Software-Produkte in Betracht, bei der die Ein-/Ausgabe-Performance (Pfadlängen, Durchsatz) eine wesentliche Rolle spielt.
Band-Verarbeitung und Remote-File-Verarbeitung werden nicht unterstützt.

> ℹ Mehrere parallele Prozesse können eine Datei gleichzeitig bearbeiten (Shared Update Mode). Die Zugriffsmethode FASTPAM bietet bei Mehrfachzugriff auf eine Datei **keine** Synchronisierungsmechanismen an. Sie müssen Sperrmechanismen, wie sie die Zugriffsmethode UPAM bietet, **selbst** implementieren.

FASTPAM-Makros und ihre Funktionen

Zur Dateibearbeitung stehen die beiden Makros FPAMSRV und FPAMACC zur Verfügung, mit denen die entsprechenden Funktionen und Operationen ausgeführt werden können.

Makro	Funktion	Kurzbeschreibung
FPAMSRV	ENABLE ENVIRONMENT	Systemumgebung für FASTPAM-Bearbeitung vorbereiten
	ENABLE IOAREA POOL	IO-Area-Bereich für FASTPAM-Bearbeitung vorbereiten
	OPEN FILE	Datei zur Bearbeitung mit FASTPAM öffnen
	CLOSE FILE	(eine mit FASTPAM geöffnete) Datei schließen; hierbei kann auf Wunsch der Last Page Pointer angegeben werden.
	DISABLE IOAREA POOL	IO-Area-Bereich für FASTPAM-Bearbeitung abbauen
FPAMACC	ACCESS FILE	(eine mit FPAMSRV) geöffnete Datei bearbeiten

Tabelle 46: FASTPAM-Makros (Funktionen)

20.1 FASTPAM-Bereiche

Um Dateiverarbeitung mit FASTPAM betreiben zu können, müssen zunächst ein FASTPAM-Environment sowie ein FASTPAM-IO-Area-Pool eingerichtet werden.
Dabei erfolgen seitens des Systems vorbereitende Validierungen sowie gegebenenfalls Fixierungen von Benutzerbereichen (Parameterliste, Ein-/Ausgabepuffer) und die Vorbereitung von Systembetriebsmitteln.

Mit der Möglichkeit, ein FASTPAM-Environment und einen FASTPAM-IO-Area-Pool resident einzurichten, verfügen der Benutzer über ein Instrument, Systembetriebsmittel für unbestimmte Zeit zu belegen. Dies ist allerdings nur mit einer besonderen Berechtigung möglich (siehe „FASTPAM-Berechtigung“ (Dateiverarbeitung mit FASTPAM)).

> ¡ In der folgenden Beschreibung zur Zugriffsmethode FASTPAM wird für den Begriff „FASTPAM-Environment“ auch der kürzere Begriff „Environment“ und für den Begriff „FASTPAM-IO-Area-Pool“ auch der Begriff „IO-Area-Pool“ verwendet.

20.1.1 FASTPAM-Environment (Speicherbereiche)

Ein FASTPAM-Environment besteht aus datei- und task-übergreifenden System- und Benutzerspeicherbereichen, die vor der Dateieröffnung angelegt und bei den Dateizugriffen immer wieder benutzt werden.

- Benutzerspeicherbereich:
  Der Benutzerspeicherbereich ist ein virtuell zusammenhängender Speicherbereich, in dem die FPAMACC-Parameterlisten (ACC=ACCESS) liegen.
  Die mit dem FPAMACC-Makro (siehe unten) generierten Parameterlisten werden für den Dateizugriff benötigt. Sie müssen in einem zusammenhängenden Bereich des virtuellen Adressraums liegen. Bei vorhandener FASTPAM-Berechtigung werden sie von FASTPAM resident gemacht. Die Anzahl der Parameterlisten bestimmt die maximale Anzahl gleichzeitiger Zugriffe auf die mit diesem Environment geöffneten Dateien.
- Systemspeicherbereiche (nur mit FASTPAM-Berechtigung):
  Zu jeder FPAMACC-Parameterliste wird vom System ein Ein-/Ausgabepfad angelegt. Der Ein-/Ausgabepfad umfasst den vom System für eine Ein-/Ausgabe benötigten residenten Klasse-3-Speicher. Seine Größe hängt von dem mit dem Operanden MAXIO-LEN angegebenen Wert ab:
  - 1 KB bei MAXIOLEN = *MINI
  - 2 KB bei MAXIOLEN = *MAXI

Attribute

Ein FASTPAM-Environment besitzt folgende unveränderliche Attribute:

- Name
- Geltungsbereich
- Adresse der FPAMACC-Parameterlisten
- Anzahl der FPAMACC-Parameterlisten
- Maximale IO Länge aller Dateizugriffe mit diesem Environment
- Eventing: Ja oder Nein, falls Ja: Kurzkennung der Ereigniskennung (Ereigniskurzkennung)

Ein FASTPAM-Environment wird durch Name und Geltungsbereich eindeutig bestimmt.Ein Environment kann für beliebig viele Dateien von beliebig vielen Tasks, die innerhalb des Geltungsbereichs des Environments liegen, genutzt werden. Der Geltungsbereich eines Environments wird durch den Speicherbereich bestimmt, in dem die FPAMACC-Parameterlisten angelegt werden:

- Liegt der Bereich im task-lokalen Adressraum, so ist das Environment nur für diese Task gültig, d.h. eine andere Task kann sich selbst mit den gleichen Parametern nicht diesem Environment anschließen.
- Liegt der Bereich in einem Common Memory Pool, so wird für das FASTPAM-Environment der Geltungsbereich übernommen, der beim Makro ENAMP (ENABLE MEMORY POOL), Operand SCOPE angegeben wurde.

20.1.2 FASTPAM-IO-Area-Pool

Ein FASTPAM-IO-Area-Pool ist ein virtuell zusammenhängender Speicherbereich, aus dem bei den Dateizugriffen die jeweiligen Ein-/Ausgabepuffer entnommen werden können. Er kann sowohl im task-lokalen Adressraum, in einem Datenraum (Data Space), als auch, bei Mehrtasksystemen, in einem Common Memory Pool liegen. In diesem Fall muss der Memory Pool allerdings für jede Task mit derselben virtuellen Adresse beginnen (Makro ENAMP, Operand FIXED=YES).

Attribute

Ein FASTPAM-IO-Area-Pool besitzt folgende unveränderliche Attribute:

- Name
- Geltungsbereich
- Adresse
- Länge

Ein FASTPAM-IO-Area-Pool wird eindeutig durch Namen und Geltungsbereich bestimmt. Analog zum FASTPAM-Environment kann ein IO-Area-Pool für beliebig viele Dateien von beliebig vielen Tasks, die im Geltungsbereich des IO-Area-Pools liegen, genutzt werden. Der Geltungsbereich eines IO-Area-Pools wird durch seine Adresse bestimmt:

- Liegt die Adresse im task-lokalen Adressraum, so ist der IO-Area-Pool nur für diese Task gültig, d.h. eine andere Task kann sich selbst mit den gleichen Parametern nicht diesem IO-Area-Pool anschließen.
- Liegt die Adresse in einem Common Memory Pool, so wird als Geltungsbereich der Bereich übernommen, der beim Makro ENAMP, Operand SCOPE, angegeben wurde.
- Liegt die Adresse in einem Datenraum (Data Space), so wird als Geltungsbereich der Bereich übernommen, der beim Einrichten des Datenraumes mit dem Operanden SCOPE angegeben wurde.

20.2 Dateiverarbeitung mit FASTPAM

Dateiformat

FASTPAM bearbeitetet nur PAM-Dateien mit den Attributen BLKCTRL=NO/DATA und BLK-SIZE=(STD,2n), wobei n=1,2,3...8 ist. Dateien, die dieses Format nicht aufweisen, müssen zunächst konvertiert werden. Dateien mit dem Attribut BLKCTRL=DATA werden mit einem Block-Kontrollfeld versehen.

FASTPAM-Berechtigung

Sie benötigen einen Eintrag im Benutzerkatalog, der Sie berechtigt, über FASTPAM-Aufrufe residenten Speicher zu erhalten. Beim Aufruf des Kommando SHOW-USER-ATTRIBUTES muss im Feld DMS-TUNING-RESOURCES der Wert *EXCLUSIVE eingetragen sein. Haben Sie diese Berechtigung nicht, so können Sie zwar mit der Zugriffsmethode FAST-PAM arbeiten, doch werden keine Bereiche resident gehalten. FASTPAM verhält sich in diesem Fall wie UPAM: vom System wird nur ein kleiner, nicht-residenter Teil der Ein-/Ausgabepfade angelegt; der Bereich der Parameterlisten und des IO-Area-Pools wird nicht fixiert. Die Folge ist, dass bei jeder Ein-/Ausgabe die Pfade neu angelegt und die Benutzerbereiche validiert und fixiert werden müssen, wodurch die FASTPAM-typischen Performance-Gewinne verloren gehen.

Für den Fall, dass keine Speicherresidenz erreicht werden kann, verhält sich FASTPAM wie bei Fehlen der FASTPAM-Berechtigung. Es wird dadurch ein dem UPAM äquivalentes oder besseres Performance-Verhalten geboten.

Speicherbereiche resident machen

Ein wesentlicher Zweck von FASTPAM besteht darin, hochperformante Dateizugriffe zu ermöglichen, indem bereits vor dem ersten Dateizugriff die notwendige Systemumgebung resident bereitgestellt wird.
Hierzu wird der Speicherbereich, der die Benutzerparameterlisten enthält, sowie der Speicherbereich, der die IO-Areas enthält (beide vom Benutzer zur Verfügung gestellt), durch das „FASTPAM-Seitenfixieren“ speicherresident gemacht.

Dies ist im Wesentlichen derselbe Vorgang, der bei anderen Zugriffsmethoden bei jeder Ein-/Ausgabe für die IO-Area von PPAM ausgeführt wird. Nur wird bei diesen anderen Zugriffsmethoden die IO-Area nach Abschluss der Ein-/Ausgabe wieder freigegeben.

Bei FASTPAM bestimmen Sie durch ENABLE/DISABLE ENVIRONMENT, wie lange die Parameterlisten fixiert sind, und durch ENABLE/DISABLE IOAREA POOL, wie lange die IO-Areas fixiert sind. Während dieser Zeit können Sie damit arbeiten. Die Validierung muss nur anfangs einmal erfolgen, da ein Freigeben fixierter Bereiche nicht möglich ist.

Außerdem wird bei ENABLE ENVIRONMENT der für die Ein-/Ausgaben benötigte Systemspeicher angefordert (je 1* pro parallel mögliche IO). Der größte Teil dieses Speichers, der von IOCTRL verwendete Bereich, ist immer resident. Dies ist auch bei anderen Zugriffsmethoden der Fall, doch wird er bei anderen Zugriffsmethoden für jede Ein-/Ausgabe neu zugeteilt und nicht permanent belegt.

Des Weiteren besteht dieser Systemspeicher aus einer FASTPAM-Workarea, die vor allem die Parameterliste zum Aufruf von PPAM enthält.

Alle diese Fixierungen erfolgen jedoch nur, wenn die Benutzerkennung über die FASTPAM-Berechtigung verfügt.

In diesem Fall – und wenn die betreffenden Fixierungen durchgeführt werden konnten – kann man von einem „residenten“ Environment bzw. einem „residenten“ IO-Area-Pool sprechen.

„Residentes Environment“ bedeutet:

- residente vorvalidierte Parameterlisten
- vorbestellter Systemspeicher
- residente FASTPAM-Workarea

„Residenter IO-Area-Pool“ bedeutet:

- residente vorvalidierte IO-Areas

Voraussetzungen für residente FASTPAM-Bereiche

- Es werden die entsprechenden Parameter angegeben
  (Makro FPAMSRV, FCT=*ENAENV/*ENAIPO, Operand RES=YES)
- der Aufrufer verfügt über die FASTPAM-Berechtigung
- es wird nicht mit Datenräumen gearbeitet
- es ist genügend freier Hauptspeicherplatz vorhanden
- die beim Programmaufruf zugeteilten residenten Seiten reichen aus (Kommando START-EXEC-PROGRAM /LOAD-PROGRAM, Operand RESIDENT-PAGES);
  Voraussetzung für die Zuteilung residenter Seiten beim Programmaufruf ist, dass die im Benutzerkatalog festgelegte Höchstgrenze und die systemglobale Grenze für residente Speicherseiten nicht überschritten wird.

20.3 Multi-User-Betrieb

Eine PAM-Datei kann parallel mit folgenden Zugriffsmethoden erstellt bzw. bearbeitet werden:

- FASTPAM
- UPAM (siehe Kapitel „UPAM – User Primary Access Method“ (UPAM - User Primary Access Method))
- DIV (siehe Kapitel „DIV – Data In Virtual“ (DIV - Data In Virtual))

Möglichkeiten der parallelen Eröffnungen

			USER B								
	SHARUPD=		***YES**			***NO**			***WEAK**		
		OPEN-Modus	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A	***YES**	INPUT INOUT OUTIN	X O O	O O O		X			X X X		
	***NO**	INPUT INOUT OUTIN	X			X			X X X		
	***WEAK**	INPUT INOUT OUTIN	X	X		X	X		X X X	X	

Tabelle 47: FASTPAM: erlaubte SHARUPD-/OPEN-Kombinationen

X: OPEN erlaubt

O: OPEN nur erlaubt,

- wenn die Eröffner dieselbe blockorientierte Zugriffsmethode benutzen (nur UPAM/FASTPAM oder nur DIV)
- **und** denselben Wert für den Operanden LOCKENV benutzen (alle LOCKENV=*HOST oder LOCKENV=*XCS)
- **und** alle im selben HOST laufen *oder* im selben XCS-Verbund bei Verwendung von LOCKENV=*XCS

Hinweise

- Leseoperationen mit SHARUPD=*WEAK können eine Datei gleichzeitig mit jeder beliebigen Schreiboperation eröffnet haben. (SHARUPD=*WEAK ist nur bei UPAM und DIV möglich.)

  Ausnahme:
  Für Leseoperationen mit DIV-SHARUPD=*WEAK, die bei OPEN LOCVIEW=*MAP angegeben haben, ist eine parallele Eröffnung mit einer UPAM/FASTPAM-Schreiboperation nicht erlaubt.

  Leseoperationen mit DIV-SHARUPD=*WEAK, die bei OPEN LOCVIEW=*NONE spezifiziert haben, besitzen dieselbe Verträglichkeit wie Leseoperationen mit UPAM/FASTPAM-SHARUPD=*WEAK.
- Eröffner mit DIV-SHARUPD=*YES sind nicht mit Eröffnern mit UPAM/FASTPAM-SHARUPD=*YES verträglich.
- Leseoperationen sind immer miteinander verträglich (unabhängig von Zugriffsmethode, SHARUPD-Spezifikation, LOCKENV-Spezifikation und Host.
- Nicht erlaubte Kombinationen führen zu einem OPEN-Fehler.
- Bei Zugriffen auf Dateien im Modus SHARUPD=YES kann der Fall eintreten, dass eine Datei mit einer Dateigröße < 32 GB durch entsprechende Verarbeitung zu einer Datei >= 32 GB wird.

  Hier werden zwei Fälle unterschieden:

  - Aufrufer, die auf diese Situation vorbereitet sind
    (mit der Angabe LARGE_FILE=*ALLOWED beim Makro FCB bzw.
    EXCEED-32GB=*ALLOWED beim Kommando ADD-FILE-LINK)
  - nicht vorbereitete Aufrufer
    (Angabe LARGE_FILE=*FORBIDDEN beim Makro FCB bzw.
    EXCEED-32GB=*FORBIDDEN beim Kommando ADD-FILE-LINK).

  Es wird nach jedem Aufruf des Allocators die Größe der betroffenen Datei überprüft. Wenn bei dieser Überprüfung eine Dateigröße >= 32 GB ermittelt wird und im zugehörigen FCB das Attribut LARGE_FILE=*FORBIDDEN gesetzt ist, wird die Verarbeitung abgebrochen und ein entsprechender Returncode in der eigenen Parameterliste FPAMSRV(I) bzw. eine DMS-Meldung ausgegeben.

Für die Zugriffsmethode FASTPAM gilt Folgendes:

- Mehrere parallele Prozesse (mehrere SHARUPD=*YES und MODE=*OUTIN/*INOUT-Eröffner) können eine Datei gleichzeitig mit FASTPAM bearbeiten.

  > **i** Bei Mehrfachzugriff auf die Datei (Shared Update Mode) muss der Benutzer selbst für entsprechende Synchronisierungsroutinen sorgen, falls diese vom eingesetzten Softwareprodukt nicht vorgesehen sind. Im Unterschied zu UPAM stellt FASTPAM hierfür keine Blocksperren-Mechanismen zur Verfügung.

- FASTPAM- und UPAM-Eröffner

  Eine Datei kann parallel von mehreren Tasks sowohl mit FASTPAM als auch mit UPAM eröffnet werden. Die Bearbeitung wird von den Operanden MODE und SHARUPD (siehe unten) der Funktion OPEN gesteuert. FASTPAM unterstützt zwar SHARUPD= *WEAK nicht, verhält sich aber ansonsten genauso wie UPAM, sowohl bei FASTPAM-Eröffnern unter sich, als auch bei gemischten UPAM- und FASTPAM-Eröffnern.

  Wird auf eine Datei gleichzeitig mit UPAM und FASTPAM zugegriffen, so muss auch der UPAM-Benutzer sich selbst mit dem FASTPAM-Benutzer synchronisieren, da die UPAM-Blocksperren nur bei beidseitiger Anwendung wirken und FASTPAM keinen Blocksperren-Mechanismus zur Verfügung stellt.

- FASTPAM- und DIV-Eröffner

  FASTPAM verhält sich zu DIV genauso wie UPAM. Eine parallele Verarbeitung ist nur erlaubt, wenn die Datei von allen mit INPUT eröffnet wird.
- Dateien auf Shared Pubsets (SPVS) werden von FASTPAM unterstützt: FASTPAM-Eröffner können von verschiedenen Systemen auf einen Shared-Pubset lesend zugreifen, auch parallel mit UPAM- und DIV-Eröffnern, die lesend zugreifen.
- Systemübergreifender Dateizugriff (RFA) wird nicht unterstützt.

20.4 Datenkonsistenz

Datenkonsistenz im Multi-User-Betrieb

Die Zugriffsmethode FASTPAM bietet bei Mehrfachzugriff auf eine Datei (Shared-Update-Mode) keine Synchronisierungsmechanismen an. Der Benutzer muss daher selbst für entsprechende Synchronisierungsroutinen sorgen, falls diese vom eingesetzten Software-Produkt nicht vorgesehen sind.
Beim gemeinsamen Shared-Update-Betrieb einer FASTPAM-, UPAM- und ggf. DIV-Anwendung muss für alle Zugriffe ein gemeinsamer Synchronisierungsmechanismus verwendet werden.

Datenkonsistenz nach einem Systemausfall

Tritt bei einem ACCESS-FILE-Auftrag ein Fehler auf, kann nicht angegeben werden, ob und wie viele Daten übertragen worden sind. Das Schreiben eines Blocks darf nicht als atomar angenommen werden. Der Inhalt der Datei kann in einem solchen Fall in einem inkonsistenten Zustand sein.

20.5 Funktionelle Unterschiede zwischen UPAM und FASTPAM

- Mit FASTPAM können ausschließlich PAM-Dateien mit folgenden Dateieigenschaften verarbeitet werden:
  - BLOCK-CONTROL-INFO=*NO oder *DATA
  - BUFFER-LENGTH = *STD(2n), n=1,2...8
- Folgende Funktionen werden von FASTPAM nicht unterstützt:
  - DUMMY-Dateien
  - Bandverarbeitung
  - RFA
- FASTPAM unterstützt synchrone und asynchrone Lese- und Schreiboperationen. Folgende von UPAM angebotene Operationen werden von FASTPAM nicht unterstützt:
  - CHK
  - LOCK / UNLOCK
  - LRD / LRDWT / WRTWU
  - SETL
  - SYNC
- Mit FASTPAM können Ein-/Ausgaben in Datenräume (Data Spaces) erfolgen.
- Die Funktionalität der UPAM-Operation SETLPP wird im Rahmen der FASTPAM-CLOSE-Verarbeitung bereitgestellt.
- Die Funktion SHARUPD=*WEAK wird nicht unterstützt (siehe auch "Multi-User-Betrieb").
- Ein impliziter WAIT ist nicht möglich.
- Innerhalb einer OPEN/CLOSE-Klammer können asynchrone Ein-/Ausgaben entweder nur durch WAIT abgeschlossen oder ihre Beendigung nur über den Eventing-Mechanismus gemeldet werden.
- Eine relative Seitennummern-Angabe ist nicht möglich.

21 ISAM - Indexed Sequential Access Method

ISAM ist wie SAM eine satzorientierte Zugriffsmethode für Plattendateien. Im Unterschied zu SAM-Dateien lassen sich ISAM-Dateien jedoch nicht nur sequenziell verarbeiten: Gesteuert über einen Schlüsselbereich im Datensatz (ISAM-Schlüssel, Primärschlüssel) werden die Sätze einer ISAM-Datei in deren Datenblöcken abgelegt. Über Indexblöcke verwaltet das DVS diese Datenblöcke, sodass die einzelnen Datensätze direkt über ihre Schlüssel auffindbar sind.

Blockformate

Es gibt zwei Ausprägungen der Zugriffsmethode ISAM, mit denen Dateien unterschiedlicher Blockformate (siehe Abschnitt „Blockformate für Plattendateien“) verarbeitet werden können:

- NK-ISAM (Nonkey-ISAM) verarbeitet Dateien des Blockformats „DATA“: Solche Dateien enthalten keine gesonderten PAM-Schlüssel. Die DVS-Verwaltungsinformation wird innerhalb der PAM-Seite in einem Blockkontrollfeld hinterlegt.
- K-ISAM (Key-ISAM) verarbeitet Dateien des Blockformates „PAMKEY“: Diese Dateien sind dadurch gekennzeichnet, dass für jede PAM-Seite DVS-Verwaltungsinformation in einem eigenen (außerhalb der Seite gelegenen) PAM-Schlüssel geführt wird.

Mit dem Operanden BLKCTRL in den Makros FILE und FCB bzw. mit dem Operanden BLOCK-CONTROL-INFO im Kommando ADD-FILE-LINK können Sie zwischen diesen beiden Verarbeitungsformen wählen:

- BLKCTRL=DATA/DATA2K/DATA4K vereinbart NK-ISAM
- BLKCTRL=PAMKEY vereinbart K-ISAM

Wenn Sie keine explite Angabe zu BLKCTRL machen, steuert der Systemparameter BLKCTRL allgemein, ob eine Datei auf K-Platten mit der Eigenschaft BLKCTRL=PAMKEY oder DATA/NO angelegt wird.
Speziell beim Anlegen von ISAM-Dateien steuert ab BS2000 OSD/BC V11.0 der Systemparameter ISBLKCTL, ob die Datei auf K-Platten als NK-ISAM-Datei angelegt wird.
Da K-ISAM in der Praxis mehr und mehr an Bedeutung verliert, ist c'NONKEY' die Voreinstellung von ISBLKCTL, d. h. standardmäßig wird eine ISAM-Datei auch auf K-Platten als NK-ISAM-Datei angelegt.

NK-ISAM bietet Ihnen alle Funktionen des K-ISAM mit weitgehend identischen Schnittstellen. Im Unterschied zu K-ISAM benutzt NK-ISAM für Schreib- und Lesezugriffe ISAM-Pools: Dies sind Bereiche im Arbeitsspeicher, in denen große Teile der zu verarbeitenden ISAM-Dateien resident gehalten werden. Durch Verringerung der Ein-/Ausgaberate können diese die Verarbeitung insbesondere bei nichtsequenziellen Zugriffen beschleunigen. (Weitere Informationen zu ISAM-Pools siehe unter „Übersicht über die wichtigsten Funktionen des NK-ISAM“).

Bei NK-ISAM-Dateien ist es darüber hinaus möglich, in den Datensätzen neben dem ISAM-Schlüssel (Primärschlüssel) bis zu 30 weitere so genannte Sekundärschlüssel zu definieren, über die Sie – ebenso wie über den Primärschlüssel – bei der Verarbeitung Datensätze mit gewünschten Schlüsselwerten auffinden können. Durch diese Funktionserweiterung können ISAM-Dateien wesentlich flexibler als mit Primärschlüsseln allein verarbeitet werden.

Bei NK-ISAM-Dateien ist zwischen NK2-ISAM-Dateien und NK4-ISAM-Dateien zu unterscheiden (siehe Abschnitt „NK4-Format“).

In diesem Kapitel wird die Zugriffsmethode ISAM beschrieben, d.h. die Funktionen, die von NK-ISAM und K-ISAM unterstützt werden. Auf Besonderheiten und Inkompatibilitäten wird an den entsprechenden Stellen verwiesen, im Abschnitt „Kompatibilität“ sind sie noch einmal zusammengefasst.

Die folgenden Abschnitte bieten zunächst eine Kurzbeschreibung der Funktionen im NK-ISAM. Im Anschluss daran wird der Dateiaufbau von NK-ISAM- und K-ISAM-Dateien beschrieben, die Verarbeitung von NK-ISAM-Dateien in ISAM-Pools, ISAM-spezifische Verarbeitungseigenschaften, Shared-Update-Verarbeitung, OPEN-Modi und Programmierhinweise für ISAM-Verarbeitung. Weitere Abschnitte enthalten Hinweise für den Einsatz von NK-ISAM und für die Umstellung von K-ISAM nach NK-ISAM sowie Hinweise zum „Crash-Verhalten“ von ISAM-Dateien.

Übersicht über die wichtigsten Funktionen des NK-ISAM

ISAM-Pools

Die Ein-/Ausgaberate wird reduziert, wenn – bei nichtsequenzieller Verarbeitung – große Teile der Datei im virtuellen Speicher resident gehalten werden. ISAM-Pools enthalten neben den Daten- und Indexblöcken auch die notwendigen Verwaltungsinformationen.
Eine Kommando- und eine Makroschnittstelle erlauben es Ihnen, selbst Pools anzulegen und zu verwalten, wodurch Sie diese den Erfordernissen Ihrer Dateiverarbeitung optimal anpassen können. Wenn Sie von dieser Möglichkeit keinen Gebrauch machen, legt NK-ISAM benutzerspezifische Standardpools an, in denen es die zu verarbeitenden Dateien puffert.

Schlüsselkomprimierung

Die Länge der Indexeinträge wird durch die Speicherung komprimierter Schlüssel klein gehalten. Dadurch erhöht sich das Fassungsvermögen der Indexblöcke.

Sperren

Externe Sperren sind als Schlüsselsperren oder bei sequenzieller Verarbeitung als Bereichssperren realisiert.

Blockmerge

Für Datenblöcke wird eine Mindestfüllung von ca. 40 %, für Indexblöcke von ca. 45 % garantiert. Unterschreitet die Füllung eines Blocks diese Grenze, wird der Blockinhalt auf benachbarte Blöcke umverteilt oder mit dem Inhalt eines benachbarten Blocks zusammengefasst.

Sekundärschlüssel

Zusätzlich zum ISAM-Schlüssel (Primärschlüssel) können Sie in den Datensätzen bis zu 30 Sekundärschlüssel definieren. Ebenso wie über seinen Primärschlüssel kann ein Satz auch über einen Sekundärschlüssel gelesen werden. Über eine Kommando- und eine Makroschnittstelle können Sie Sekundärschlüssel vereinbaren, löschen und sich über sie informieren. Für den Zugriff auf Datensätze über Sekundärschlüssel stellt NK-ISAM Operanden in den Makros für Lese- und Zeigeroperationen zur Verfügung.

21.1 ISAM-Dateistrukturen

ISAM-Dateien bestehen nicht nur aus Datenblöcken, sondern z.B. auch aus Blöcken, die Verwaltungsinformationen für die ISAM-Verarbeitung enthalten. NK-ISAM-Dateien bestehen aus Datenblöcken, Indexblöcken, Kontrollblock, freien Blöcken und u.U. Überlaufblöcken, K-ISAM-Dateien aus Datenblöcken, Indexblöcken und freien Blöcken. Da Kontrollblock und Überlaufblöcke von ISAM angelegt und verwaltet werden, ergeben sich an der Benutzerschnittstelle keine Unterschiede bei der Verarbeitung von NK-ISAM- oder K-ISAM-Dateien.

In den folgenden Abschnitten werden die Strukturelemente einer ISAM-Datei beschrieben, ausgehend von den Datensätzen des Benutzers, die in den Datenblöcken „abgelegt“ werden und bei NK-ISAM in manchen Fällen in sog. „Überlaufblöcke“ hineinragen.
Bei NK-ISAM werden die Datenblöcke über die Indexblöcke und den Kontrollblock verwaltet, bei K-ISAM nur über die Indexblöcke.

21.1.1 Dateibereiche

Datenblöcke, Indexblöcke, Index-Datenblöcke, freie Blöcke sowie Kontrollblock und Überlaufblöcke bilden den genutzten Dateibereich einer ISAM-Datei.
Als ungenutzter Dateibereich wird dagegen der Teil einer Datei bezeichnet, den der Benutzer jederzeit freigeben kann (mit dem SPACE-Operanden im Makro FILE bzw. im Kommando MODIFY-FILE-ATTRIBUTES).

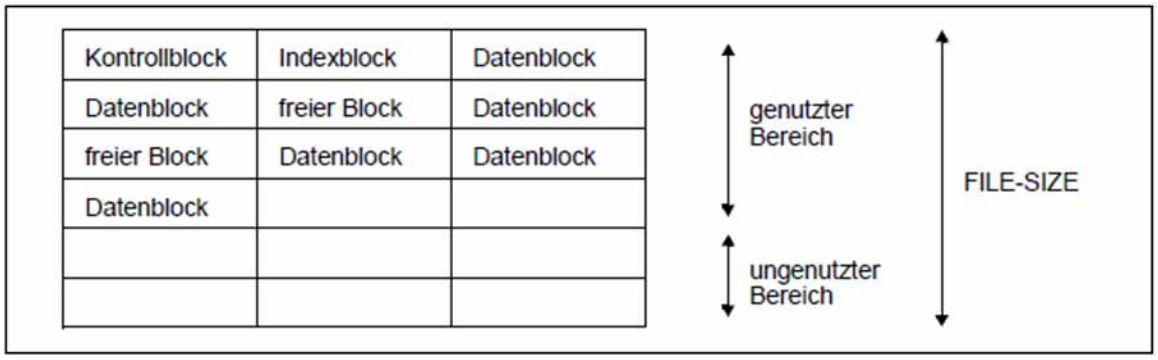


Bild 27: Dateibereiche von ISAM-Dateien

Jedes Feld in diesem Bild entspricht einer PAM-Seite, leere Felder zeigen an, dass diese PAM-Seiten von ISAM bisher nicht genutzt wurden.

FILE-SIZE	Gesamtgröße der ISAM-Datei (hier: 18 PAM-Seiten)
genutzter Bereich	alle bereits einmal für die Datei genutzten Blöcke, auch wieder freie Blöcke (hier: HIGH-US-PA = 10)
ungenutzter Bereich	von ISAM bisher nicht genutzte Blöcke, freigebbar (8 PAM-Seiten)

21.1.2 Datensatz ohne Sekundärschlüssel

ISAM-Datensätze zeichnen sich durch einen Indexbereich aus, der in allen Sätzen einer Datei an der gleichen Stelle steht und gleich strukturiert ist. Der Indexbereich enthält immer einen Satzschlüssel, der den Satz identifiziert. Daneben kann der Index auch Markierungen enthalten, die ein Durchsuchen der Datensätze nach bestimmten Merkmalen ermöglichen. Der gesamte Indexbereich (Schlüssel + Markierungen) darf nicht länger als 255 Bytes sein.

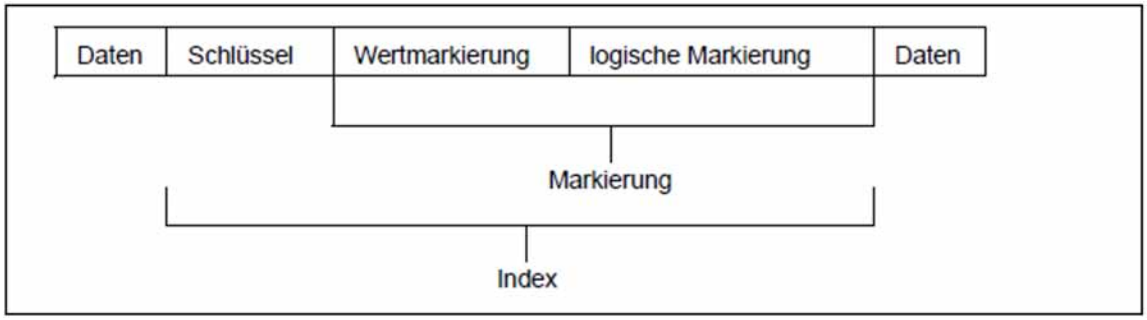


Bild 28: Aufbau des ISAM-Index

ISAM-Schlüssel

Jeder Satz einer ISAM-Datei muss einen mindestens ein Byte langen Schlüssel besitzen. Die Position des ISAM-Schlüssels im Datensatz wird für alle Sätze der Datei in den Makros FILE und FCB (Operand KEYPOS) bzw. im Kommando ADD-FILE-LINK (Operand KEY-POSITION) festgelegt. Bei der Definition der Schlüsselposition müssen Sie das Satzformat (siehe unter „ISAM-Satzformate“ (Datensatz mit Sekundärschlüsseln (NK-ISAM))) berücksichtigen.
So kann z.B. in Dateien mit variablem Satzformat der ISAM-Schlüssel erst ab der Position 5 beginnen (KEYPOS >= 5) wegen der 4 Bytes für Satzlängenfeld und Steuerinformationen.

Die Schlüssellänge wird durch den Operanden KEYLEN in FILE/FCB bzw. durch den Operanden KEY-LENGTH in ADD-FILE-LINK festgelegt. Bei der Definition der Schlüssellänge müssen Sie darauf achten, dass die maximale Indexlänge von 255 Bytes für Schlüssel plus Markierungen nicht überschritten werden darf.

Bei K-ISAM kann sich die Schlüssellänge auf die Dateigröße und auf die Performance auswirken. Da der gesamte Schlüssel in den Indexblock aufgenommen wird, hängt von der Schlüssellänge ab, wie viele Indexeinträge in einen Indexblock aufgenommen werden können.

Bei NK-ISAM optimiert das DVS durch Schlüsselkomprimierung die Anzahl der Indexeinträge im Indexblock, sodass längere Schlüssel auf im Durchschnitt 3-5 Bytes komprimiert werden können und jeder Indexblock durchschnittlich 160 Einträge enthält. Die Anzahl Indexeinträge pro Block wirkt sich wiederum aus auf die Gesamtzahl der Indexblöcke und damit auf die Anzahl der Indexstufen einer Datei und die Performance beim Zugriff auf einen Satz.

Mehrfache ISAM-Schlüssel (Duplicate Key)

Das Prinzip der ISAM-Dateiverarbeitung basiert auf eindeutigen Satzschlüsseln. Mit der Angabe DUPEKY=YES in FILE/FCB bzw. mit DUPLICATE-KEY im Kommando ADD-FILE-LINK können Sie Dateien erzeugen, in denen Satzschlüssel mehrfach auftreten können.

Beim Schreiben werden Sätze mit bereits in der Datei enthaltenen Schlüsseln stets hinter den letzten Satz mit dem gleichen Schlüssel in die Datei eingefügt. Die Anordnung von Sätzen mit gleichen Schlüsseln in einer Datei entspricht also der zeitlichen Folge, in der sie in die Datei eingefügt wurden.

NK-ISAM fügt Sätzen mit Schlüsseln, die bereits in der Datei vorkommen, einen Zeitstempel an. Durch den Zeitstempel wird die korrekte Reihenfolge der Sätze in der Datei, entsprechend dem Zeitpunkt ihrer Aufnahme, gesichert.

Sequenzielle Leseoperationen stellen die Sätze in der Reihenfolge bereit, in der sie geschrieben wurden, beim „Rückwärtslesen“ wird die Reihenfolge dementsprechend umgekehrt.

Nichtsequenzielles Lesen über den Satzschlüssel liefert immer den ersten Satz einer Satzfolge mit gleichen Schlüsseln. Die weiteren Sätze dieser Folge müssen dann sequenziell gelesen werden.

Wird bei Shared-Update-Verarbeitung von NK-ISAM-Dateien eine Satzsperre wirksam, umfasst diese Sperre alle Sätze mit dem gleichen Schlüssel.

> ¡ Enthält eine ISAM-Datei viele Sätze mit gleichen Schlüsseln, kann dies zu erheblichen Performance-Verlusten führen. Bei NK-ISAM-Dateien mit sehr langen Sätzen ist darauf zu achten, dass durch das Anfügen des Zeitstempels Überlaufblöcke entstehen können.

Markierungen im ISAM-Index

Neben dem ISAM-Schlüssel können auch Wertmarkierungen oder logische Markierungen zur Suche von Sätzen verwendet werden. Diese Markierungen (Flags) folgen im ISAM-Index auf den Schlüssel, und zwar Wertmarkierung vor logischer Markierung, wenn beide Markierungsarten genutzt werden. Die Markierungen werden bei NK-ISAM nicht in den Indexeintrag übernommen, sondern intern sequenziell ausgewertet. Für den Benutzer ergeben sich bei der Verwendung markierter Sätze daher Performanceverluste, ansonsten erfolgt die Verarbeitung markierter Sätze wie bei K-ISAM.

> ¡ Bei NK-ISAM erfolgt die Flagverarbeitung im Gegensatz zu K-ISAM mit erheblich niedrigerer Performance.

Mit K-ISAM werden die Markierungen in die Indexeinträge übernommen und setzen sich – als Bestandteil des ISAM-Index – in allen Indexstufen der Datei fort. Die Übernahme der Markierung in den Indexeintrag verläuft bei Wert- und logischer Markierung jedoch unterschiedlich.

Die Position der Wertmarkierung ist bestimmt durch Position und Länge des vorausgehenden Schlüssels (KEYPOS + KEYLEN), die Länge wird mit VALLEN in den Makros FILE ind FCB bzw. mit dem Operanden VALUE-FLAG-LENGTH im Kommando ADD-FILE-LINK festgelegt. Für die logische Markierung gilt entsprechend: die Position ist durch die Summe von Schlüsselposition, Schlüssellänge und Länge der Wertmarkierung bestimmt, die Länge durch den Operanden LOGLEN in den Makros FILE und FCB bzw. durch den Operanden LOGICAL-FLAG-LENGTH im Kommando ADD-FILE-LINK.

Bei der Suche eines Satzes über Wert- oder logische Markierung wird jeweils der Inhalt des markierten Bereichs der geprüften Sätze mit einem definierten Wert bzw. einer „logischen“ Bit-Maske verglichen. Das DVS stellt den Satz zur Verfügung, der als Erster den Bedingungen genügt.

Bei K-ISAM wird die Wertinformation der Indexeinträge auf Grund der vom Benutzer gemachten Angabe zu VALPROP in den Makros FILE und FCB bzw. zu PROPAGATE-VALUE-FLAG im Kommando ADD-FILE-LINK erstellt. Wird die Funktion MAX (MIN) angegeben, enthält jeder Indexeintrag die höchste (niedrigste) Wertmarkierung des Blockes der nächstniedrigeren Stufe (Daten- oder Indexblock), auf den er verweist. Ob die Weitergabe der maximalen oder minimalen Wertmarkierung sinnvoll ist, hängt ab von der Art, in der die Sätze später bereitgestellt werden sollen. Der Nutzen einer Wertmarkierung ist umso größer, je gleichmäßiger die Wertmarkierungen mit aufsteigendem ISAM-Schlüssel steigen oder fallen.

In der logischen Markierung können duale Eigenschaften des im Datensatz beschriebenen Objektes bitweise verschlüsselt werden. Alle logischen Markierungen eines Datenblocks (oder Indexblocks) werden mit der Funktion ODER verknüpft und in den (nächsthöheren) Indexeintrag weitergegeben. Das DVS kann nun bei der Suche nach Sätzen mit bestimmten Eigenschaften bereits nach Überprüfen des Indexeintrags entscheiden, ob in dem Block, auf den verwiesen wird, Sätze dieser Eigenschaft(en) vorhanden sind oder nicht.

Sind die Angaben zu VALLEN (bzw. VALUE-FLAG-LENGTH) oder LOGLEN (bzw. LOGICAL-FLAG-LENGTH) ungleich null, wird vorausgesetzt, dass es sich um eine Datei mit Markierungen handelt.
Existiert die Datei bereits (d.h. bei OPEN INPUT/INOUT/EXTEND), werden die Katalogwerte von VALPROP (bzw. PROPAGATE-VALUE-FLAG) oder VALLEN (bzw. VALUE-FLAG-LENGTH) verwendet. LOGLEN (bzw. LOGICAL-FLAG-LENGTH) muss immer angegeben werden, wenn die Datei mit einem Markierungsbereich erstellt wurde. Die Katalogwerte können nicht mit den Makros FILE / FCB oder dem Kommando ADD-FILE-LINK geändert werden.

Beispiel: Wertmarkierung

In einer Personaldatei soll das Geburtsjahr als Wertmarkierung dienen; im Katalogeintrag wurde VALPROP=MIN vereinbart. Das bedeutet: aus jedem Datenblock wird bei K-ISAM das niedrigste Geburtsjahr der darin enthaltenen Mitarbeiter-Datensätze in den Indexblock übernommen.

- geeignete Suchfrage: Welche Mitarbeiter sind vor 1950 geboren ? – Bei der Suche können ganze Datenblöcke auf Grund der Wertmarkierung sofort übergangen werden.
- ungeeignete Suchfrage: Welche Mitarbeiter sind nach 1950 geboren? – Es müssen alle Sätze jedes Blockes geprüft werden. Hier wäre eine Datei mit VALPROP=MAX sinnvoll.

ISAM-Satzformate

ISAM-Dateien können Sätze fester oder variabler Länge enthalten, d.h. mit Satzformat F oder V. Das Satzformat wird festgelegt im RECFORM-Operanden von FCB und FILE bzw. mit dem Operanden RECORD-FORMAT im Kommando ADD-FILE-LINK.

Gilt RECFORM=V, müssen Sie bei der Definition von Schlüsselposition und maximaler Satzlänge berücksichtigen, dass den Daten ein 4 Byte langes Feld vorangestellt wird. Bei Sätzen fester Länge (RECFORM=F) brauchen Sie dieses 4-Byte-Feld bei der Definition der Schlüsselposition nicht zu beachten. Intern stellt ISAM jedoch auch diesen Sätzen ein 4-Byte-Feld voran.

In der nachfolgenden Tabelle sind die Auswirkungen des Satzformats auf die per Makro bzw. Kommando definierte Satzlänge dargestellt.

Operand in FILE / FCB	Operand in ADD-FILE-LINK	Bedeutung von RECSIZE bzw. RECORD-SIZE
RECFORM=F	RECORD-FORMAT= *FIXED	Gibt die für alle Sätze gültige Länge in Bytes an
RECFORM=V	RECORD-FORMAT= *VARIABLE	Gibt die maximale Länge des Datensatzes an; standardmäßig gilt dann RECSIZE=BLKSIZE bzw. RECORD-SIZE=BUFFER-LENGTH

Tabelle 48: Auswirkungen des Satzformats auf die Satzlänge bei ISAM

Wenn (explizit oder implizit) annähernd gilt RECSIZE=BLKSIZE, können bei NK-ISAM Überlaufblöcke entstehen (siehe Abschnitt „Überlaufblock (NK-ISAM)“), was zu einer erhöhten Ein-/Ausgaberate führen kann.

Wird beim Zugriff auf existente Dateien RECSIZE bzw. RECORD-SIZE zu klein gewählt, kann dies Auswirkungen auf Leseoperationen haben: Liest das DVS einen längeren Satz, überträgt es nur die RECSIZE entsprechende Anzahl Bytes in das vom Benutzer definierte Empfangsfeld und bricht das Lesen mit einer Fehlermeldung ab. Der Benutzer kann dies in seinem Programm über entsprechende Fehlerroutinen abfangen.

21.1.3 Datensatz mit Sekundärschlüsseln (NK-ISAM)

In den Sätzen einer NK-ISAM-Datei können neben dem ISAM-Schlüssel (Primärschlüssel) bis zu 30 Sekundärschlüssel definiert werden, über die der Benutzer die gewünschten Datensätze auffinden kann.
Im Unterschied zu herkömmlichen ISAM-Dateien darf eine Datei mit Sekundärschlüsseln weder Wertmarkierungen noch logische Markierungen enthalten. Diese Zusätze zum ISAM-Schlüssel sind durch die Sekundärschlüssel ohnehin entbehrlich geworden.
Die Länge des Primärschlüssels ist auf 255 Byte begrenzt, die Länge jedes Sekundärschlüssels auf 127 Byte.
Die Sekundärschlüssel dürfen sich untereinander und mit dem Primärschlüssel beliebig überlappen, sodass ein Datensatz mit Sekundärschlüsseln z.B. folgenden Aufbau haben kann:

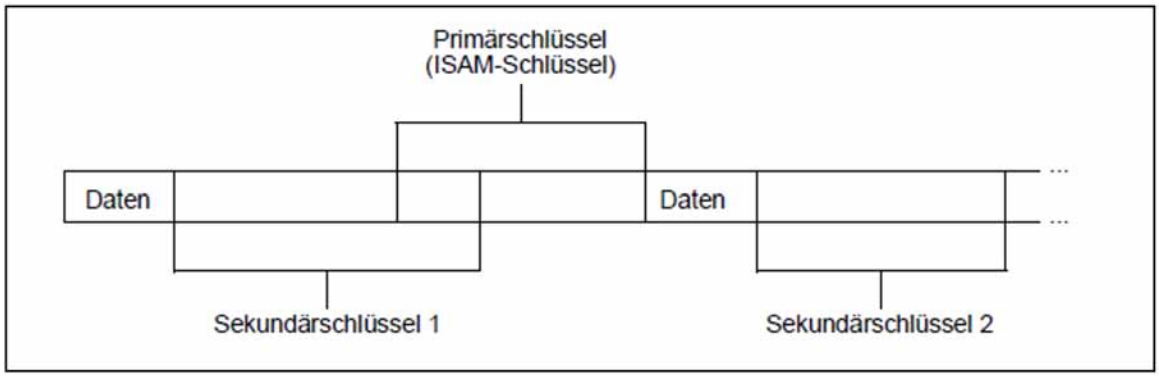


Bild 29: ISAM-Datensatz mit Sekundärschlüsseln

Primärschlüssel (ISAM-Schlüssel)

Jeder Satz einer ISAM-Datei muss einen mindestens ein Byte langen Primärschlüssel besitzen. Die Position des Primärschlüssels im Datensatz wird für alle Sätze der Datei im Makros FILE und FCB mit dem Operanden KEYPOS, im Kommando ADD-FILE-LINK mit dem Operanden KEY-POSITION festgelegt. Bei der Definition der Schlüsselposition müssen Sie das Satzformat (siehe unter „ISAM-Satzformate“) berücksichtigen. So kann z.B. in Dateien mit variablem Satzformat der ISAM-Schlüssel erst ab der Position 5 beginnen (KEYPOS >= 5 bzw. KEY-POSITION >= 5) wegen der 4 Bytes für Satzlängenfeld und Steuerinformationen.

Die Schlüssellänge wird durch den Operanden KEYLEN im Makro FILE/FCB bzw. durch den Operanden KEY-LENGTH im Kommando ADD-FILE-LINK festgelegt. Die maximale Schlüssellänge beträgt 255 Byte. Logische Markierungen und Wertmarkierungen sind nicht zulässig.

Das DVS optimiert durch Schlüsselkomprimierung die Anzahl der Indexeinträge im Indexblock, sodass längere Schlüssel auf im Durchschnitt 3-5 Bytes komprimiert werden können und jeder Indexblock durchschnittlich ca. 160 Einträge enthält.

Die Anzahl der Indexeinträge pro Block wirkt sich wiederum aus auf die Gesamtzahl der Indexblöcke und damit auf die Anzahl der Indexstufen einer Datei und die Performance beim Zugriff auf einen Satz.

Sekundärschlüssel

Zusätzlich zum Primärschlüssel darf ein Datensatz bis zu 30 Sekundärschlüssel enthalten. Wie der Primärschlüssel hat auch jeder Sekundärschlüssel in allen Sätzen einer Datei die gleiche Position und Länge.

Anders als beim Primärschlüssel werden die Merkmale eines Sekundärschlüssels (Position, Länge, Zulässigkeit mehrfach auftretender Schlüsselwerte) jedoch nicht im Katalogeintrag, sondern im Kontrollblock der Datei geführt und können daher auch nicht mit dem Makro FSTAT oder dem Kommando SHOW-FILE-ATTRIBUTES ausgegeben werden.
Zur Information über die Sekundärschlüssel einer Datei stellt ISAM deshalb den Makro SHOWAIX und das Kommando SHOW-INDEX-ATTRIBUTES zur Verfügung.

Ein weiterer Unterschied zwischen dem Primärschlüssel und einem Sekundärschlüssel besteht darin, dass der Sekundärschlüssel über einen Namen identifiziert wird. Dies ist notwendig, weil für eine Datei mehrere Sekundärschlüssel vereinbart werden dürfen (während es nur einen Primärschlüssel gibt).

Sekundärschlüssel können nur für eine existierende Datei (d.h. für eine Datei, die bereits mindestens einmal OUTPUT oder OUTIN eröffnet war) definiert werden. Zwar darf die Datei auch leer sein, doch ist es aus Performancegründen zu empfehlen, Sekundärschlüssel für eine neu zu erstellende Datei erst einzurichten, nachdem sie mit Datensätzen geladen worden ist.

Sie können einen Sekundärschlüssel mit dem Makro CREAIX oder dem Kommando CREATE-ALTERNATE-INDEX für eine Datei vereinbaren.

Die Position des Sekundärschlüssels wird mit dem Operanden KEYPOS im Makro bzw. mit dem Operanden KEY-POSITION im Kommando festgelegt. Wie beim Primärschlüssel sind auch hier bei Sätzen variabler Länge 4 Byte für Satzlängenfeld und Steuerinformation zu berücksichtigen (siehe unter „Primärschlüssel (ISAM-Schlüssel)“).

Die Länge des Sekundärschlüssels wird mit dem Operanden KEYLEN im Makro bzw. mit dem Operanden KEY-LEN im Kommando festgelegt. Ein Sekundärschlüssel darf bis zu 127 Byte lang sein. Neben der Anzahl wirkt sich auch die Länge der vereinbarten Sekundärschlüssel auf die Dateigröße und die Performance aus: Da der gesamte Schlüssel (ohne Komprimierung) in den Sekundärindexblock aufgenommen wird, hängt von der Schlüssellänge ab, wie viele Indexeinträge in einem Sekundärindexblock untergebracht werden können.

Mit dem Operanden KEYNAME im Makro bzw. KEY-NAME im Kommando wird dem Sekundärschlüssel ein Name zugeordnet. Mit diesem Namen können Sie sich in den Makros GET, GETR, GETKY und SETL auf ihn beziehen und so den Sekundärschlüssel (analog dem Primärschlüssel) für Lese- und Zeigeroperationen nutzen. Außerdem dient der Name im Makro DELAIX oder im Kommando DELETE-ALTERNATE-INDEX dazu, den Sekundärschlüssel anzugeben, der gelöscht werden soll. Über alle in einer Datei definierten Sekundärschlüssel können Sie sich mit dem Makro SHOWAIX oder mit dem Kommando SHOW-INDEX-ATTRIBUTES informieren.

Mehrfache Sekundärschlüssel (Duplicate Key)

In Dateien mit Sekundärschlüsseln müssen die Werte des Primärschlüssels eindeutig sein. Sekundärschlüsselwerte können dagegen mehrfach auftreten, sofern Sie dies nicht durch die Angabe DUPKEY=NO im Makro CREAIX bzw. DUPLICATE-KEY=*NO im Kommando CREATE-ALTERNATE-INDEX ausschließen.

Intern legt das DVS für jeden in der Datei vereinbarten Sekundärschlüssel einen oder mehrere Sekundärindexblöcke an, die für jeden in der Datei vorkommenden Wert dieses Sekundärschlüssels einen Eintrag aufnehmen, der auf den Primärschlüssel des zugehörigen Datensatzes verweist (Näheres siehe Abschnitt „Sekundärindexblock (NK-ISAM)“ (Primärindexblock (ISAM-Indexblock) und Sekundärindexblock)). Kommt ein Wert eines Sekundärschlüssels in mehreren Datensätzen vor, so enthält der zugehörige Eintrag im Sekundärindexblock auch mehrere Verweise auf die Primärschlüsselwerte dieser Datensätze. Diese Verweise sind in der Reihenfolge ihres Entstehens sortiert, d.h in der Reihenfolge, in der sie in den Sekundärindexblock geschrieben oder dort verändert wurden. (Zur internen Verwaltung fügt NK-ISAM jedem solchen Verweis auf einen Primärschlüssel zum Zeitpunkt seiner Entstehung einen Zeitstempel an.)

Sequenzielle Leseoperationen über einen Sekundärschlüssel stellen demnach die Sätze in der Reihenfolge bereit, in der die Verweise auf die zugehörigen Primärschlüsselwerte entstanden sind, d.h. in den Sekundärindexblock geschrieben oder dort verändert wurden. Beim „Rückwärtslesen“ ist die Reihenfolge umgekehrt.

Nichtsequenzielles Lesen über einen Sekundärschlüssel liefert immer den ersten Satz einer Satzfolge mit gleichen Sekundärschlüsselwerten. Die weiteren Sätze dieser Folge müssen dann sequenziell gelesen werden.

Markierungen im ISAM-Index

In Dateien mit Sekundärschlüsseln ist weder Wert- noch logische Markierung zulässig.

ISAM-Satzformate

Für Datensätze mit Sekundärschlüsseln gelten die gleichen Hinweise wie für Datensätze ohne Sekundärschlüssel (siehe "Datensatz ohne Sekundärschlüssel").

Bei der nachträglichen Vereinbarung von Sekundärschlüsseln für nichtleere Dateien mit RECFORM=V bzw. mit RECORD-FORMAT=*VARIABLE ist bei der Wahl der Schlüsselpositionen und -längen jedoch zusätzlich darauf zu achten, dass auch der kürzeste Datensatz in der Datei den zu definierenden Sekundärschlüssel noch vollständig enthalten muss.

21.1.4 ISAM-Datenblock

Die Datensätze einer ISAM-Datei werden nach ihrem ISAM-Schlüssel (Primärschlüssel) sortiert in Datenblöcken abgelegt. Diese Datenblöcke können sich aus 1-16 benachbarten PAM-Seiten zusammensetzen; aus mehr als einer PAM-Seite bestehende Datenblöcke werden auch als Multiblöcke bezeichnet.

Die Länge eines Datenblocks wird in den Makros FILE/FCB mit dem Operanden BLKSIZE, im Kommado ADD-FILE-LINK mit dem Operanden BUFFER-LENGTH festgelegt.
Da ISAM-Dateien Plattendateien sind, kann nur ein Blockungsfaktor „n“ für Standardblöcke (1 <= n <= 16) angegeben werden, keine absolute Länge. Bei der Wahl der Blocklänge muss die Verträglichkeit mit der Satzlänge berücksichtigt werden, damit der Speicherplatz optimal genutzt werden kann und keine Überlaufblöcke entstehen. Der Abschnitt „Überlaufblock (NK-ISAM)“ enthält Tabellen, die diesen Zusammenhang darstellen.

Blocksplitting – Teilen von Datenblöcken

Wenn bei nichtsequenzieller Dateierweiterung (STORE, INSRT) ein Satz in einen Block nicht mehr eingefügt werden kann, kommt es zum Blocksplitting: der alte Datenblock wird geteilt, die entstandenen Hälften werden in neue (leere) Blöcke übertragen. Der alte Datenblock bleibt der Datei zugeordnet und wird als freier Datenblock gekennzeichnet.

Beispiel: Blocksplitting

Dateistruktur

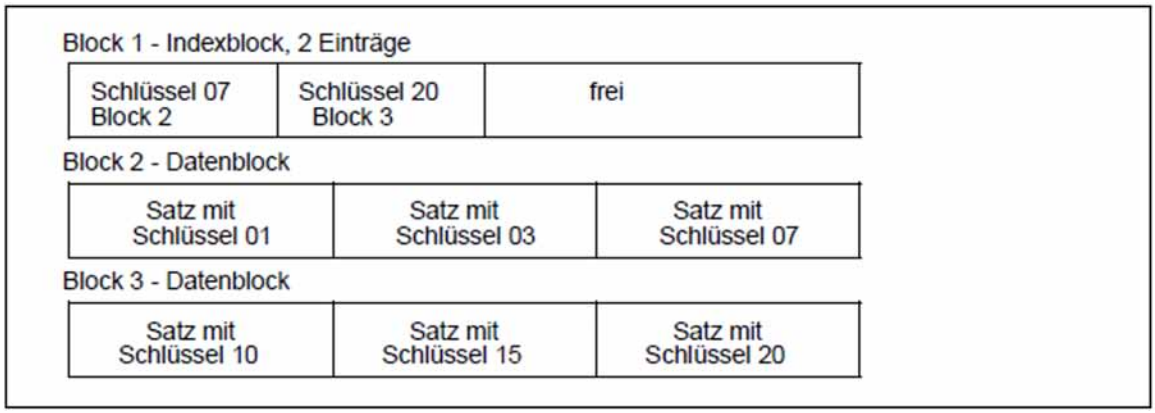


Bild 30: Beispiel einer ISAM-Dateistruktur vor Einfügen eines Satzes

Der Benutzer gibt an, dass ein Satz mit dem Schlüssel 05 eingefügt werden soll. Da kein Platz mehr im Block 2 vorhanden ist, wird Block 2 geteilt.

Die Datei hat nun folgende Struktur:

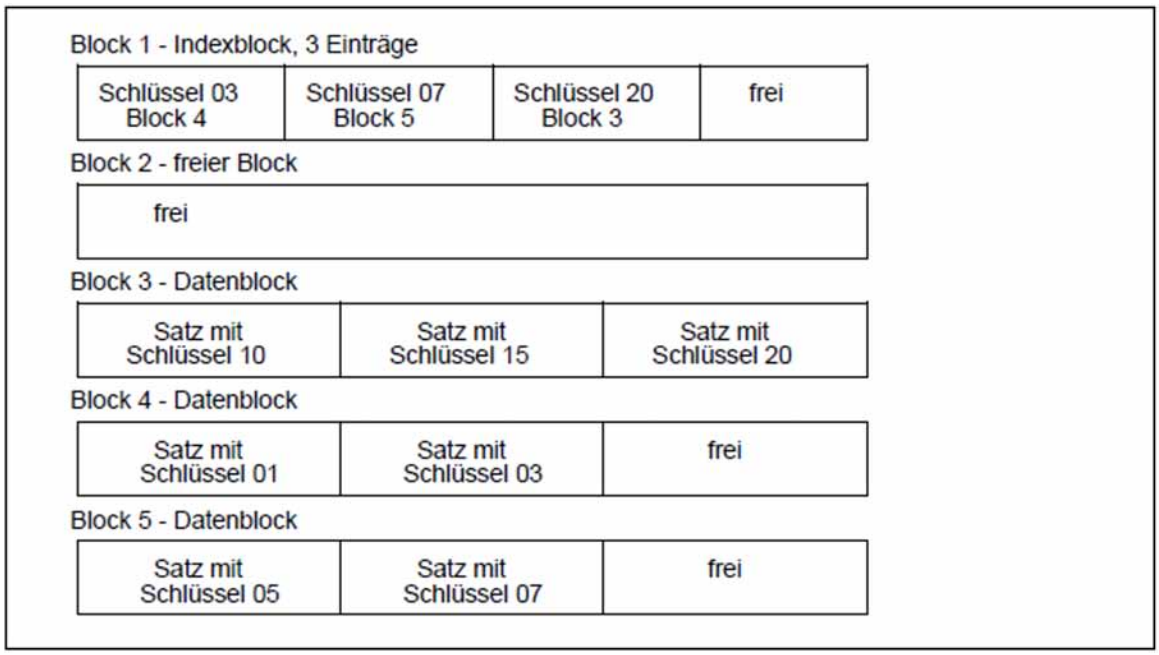


Bild 31: Beispiel einer ISAM-Dateistruktur nach Einfügen eines Satzes

21.1.5 Überlaufblock (NK-ISAM)

Überlaufblöcke sind Verlängerungen von Datenblöcken. Sie entstehen, wenn Sätze geschrieben werden, die länger sind als der nutzbare Bereich des Datenblocks. Dies ist z.B. der Fall, wenn Sie Sätze mit gleichen Schlüsseln verwenden und bei der Definition von Satz- und Blocklänge nicht berücksichtigen, dass NK-ISAM solchen Sätzen intern einen 8 Byte langen Zeitstempel anhängt. Überlaufblöcke entstehen auch bei der Umsetzung von K-ISAM-Dateien in NK-ISAM-Dateien, wenn in der K-ISAM-Datei die maximale Satzlänge ausgenutzt wurde.
Die „überlaufenden“ Daten stehen immer am Beginn des Überlaufblocks; am Ende des Blocks befindet sich ein Zeiger auf den zugehörigen Datenblock. Der Datenbereich des Überlaufblocks darf weder Teile des ISAM-Schlüssels (Primärschlüssels) noch der Markierung noch eines Sekundärschlüssels enthalten.

maximale Satzlänge / nutzbare Blocklänge

Werden folgende Regeln beachtet, entstehen keine Überlaufblöcke:

- die maximal zulässige Satzlänge ergibt sich aus der BLKSIZE- bzw. BUFFER-LENGTH-Angabe:
  n * 2048, 1 <= n <= 16 (für BLKSIZE=(STD,n) bzw. BUFFER-LENGTH=*STD(SIZE=n))
- die nutzbare Blocklänge reduziert sich um den für Verwaltungsinformationen und Zeitstempel benötigten Bereich
- da ISAM intern allen Datensätzen ein 4-Byte-Feld mit der Satzlänge voranstellt, reduziert sich bei Sätzen mit RECFORM=F bzw. RECORD-FORMAT=*FIXED die nutzbare Blocklänge noch einmal um 4 Byte

Es ergibt sich folgende Formel für die nutzbare Blocklänge:

```
RECSIZE <= n * 2032 - 16 - T - F
```

n	Blockungsfaktor aus BLKSIZE = (STD,n), 1 <= n <= 16 bzw. BUFFER-LENGTH=*STD(SIZE=n)
T	Zeitstempel
	T = 8 für Dateien mit Mehrfachschlüsseln
	T = 0 für Dateien ohne Mehrfachschlüssel
F	Satzformat
	F = 0 bei RECFORM = V bzw. RECORD-FORMAT=*VARIABLE
	F = 4 bei RECFORM = F bzw. RECORD-FORMAT=*FIXED

Da jeder Datenblock mindestens einen Satz enthält, müssen bei der Definition der Satzlänge 16 Byte für die internen Verwaltungsdaten berücksichtigt werden. Die folgende Tabelle zeigt für ISAM-Dateien mit Standardeigenschaften (Satzformat V, keine Mehrfachschlüssel) den Zusammenhang zwischen maximal zulässiger Satzlänge und nutzbarer Blocklänge für NK-ISAM-Dateien ohne Überlaufblöcke.

Blockungsfaktor n (BLKSIZE=(STD,n))	max. Satzlänge = BLKSIZE (in Byte)	nutzbare Blocklänge (in Byte)
1	2048	2016
2	4096	4048
3	6144	6080
4	8192	8112
5	10240	10144
6	12288	12176
7	14336	14208
8	16384	16240
9	18432	18272
10	20480	20304
11	22528	22336
12	24576	24368
13	26624	26400
14	28672	28432
15	30720	30464
16	32768	32496

Tabelle 49: nutzbare Blocklänge für NK-ISAM-Dateien

Maximale Schlüsselposition und -länge

Da weder ISAM-Schlüssel noch Markierungen in einen Überlaufblock hineinragen dürfen, ergeben sich auch für Schlüsselposition (KEYPOS), Schlüssellänge (KEYLEN), Wertmarkierung (VALLEN) und logische Markierung (LOGLEN) Maximalwerte – abhängig von Blockungsfaktor und Satzformat. Die zulässigen Höchstwerte lassen sich nach folgenden Formeln aus der nutzbaren Blocklänge (siehe Tabelle 49) errechnen:

ISAM-Schlüssellänge / -position:

```
KEYPOS <= SL - F - KEYLEN - VALLEN - LOGLEN + 1

KEYLEN <= SL - F - KEYPOS - VALLEN - LOGLEN + 1
```

KEYPOS	Schlüsselposition
KEYLEN	Schlüssellänge
VALLEN	Wertmarkierung (Länge)
LOGLEN	logische Markierung (Länge)
SL	nutzbare Blocklänge
F	abhängig vom Satzformat: • F = 0 für RECFORM=V • F = 4 für RECFORM=F

Beispiel

Dateieigenschaften: BLKSIZE = (STD,1)
KEYLEN = 12

Für KEYPOS ergeben sich folgende Werte:

```
RECFORM = V => KEYPOS <= 2016 - 12 + 1 = 2005
RECFORM = F => KEYPOS <= 2016 - 4 - 12 + 1 = 2001
```

21.1.6 NK4-Format

Zur Unterstützung von NK4-Datenträgern wird das Blockformat NK4-ISAM eingeführt:

Ein logischer Datenblock besteht aus einer Folge von 4-KB-Blöcken; diese entsprechen der Transporteinheit für NK4-Datenträger. Jeder 4-KB-Block beginnt mit einem Blockkontrollfeld, das aus einem 12 Byte langen allgemeinen Teil und aus einem 4 Byte langen ISAM-spezifischen Teil besteht.

Ein Datenblock besteht aus einer Folge von mindestens einem bis zu maximal acht 4-KB-Blöcken. Die Größe des logischen Datenblöcks wird durch eine Angabe im FCB- oder FILE-Makro oder im Kommando ADD-FILE-LINK gesteuert. Der Aufbau eines NK4-ISAM-Datenblocks entspricht dem eines NK2-ISAM-Datenblocks. Die Adressierung der Sätze und die Verwaltung des freien Platzes erfolgt über den Blocktrailer.

maximale Satzlänge / nutzbare Blocklänge

Die Bruttolänge eines NK4-Datenblocks ist nicht voll für Sätze des Benutzers nutzbar, da für jede 4-KB-Seite des Blocks 16 Byte durch das System für das Blockkontrollfeld genutzt werden; weitere 16 Byte pro Block sind für den Blocktrailer vorzusehen. Die effektive Nutzlänge eines NK4-Datenblocks einer Datei mit der Blockgröße (STD,n) mit n gerade ist daher:

- die maximal zulässige Satzlänge ergibt sich aus der Blocklängen-Angabe (Kommando ADD-FILE-LINK, Operand BUFFER-LENGTH; Makro FILE, Operand BLKSIZE): n * 2048, n = 2, 4, 6, 8
- die nutzbare Blocklänge reduziert sich um den für Verwaltungsinformationen und Zeitstempel benötigten Bereich (Für jeden 4-KB-Block des logischen Blocks werden 16 Byte durch das System für das Blockkontrollfeld genutzt; weitere 16 Byte pro Block sind für den Blocktrailer vorzusehen).
- da ISAM intern allen Datensätzen ein 4-Byte-Feld mit der Satzlänge voranstellt, reduziert sich bei Sätzen mit RECFORM=F die nutzbare Blocklänge noch einmal um 4 Byte (jeder Datenblock muss mindestens einen Satz enthalten).

Da die maximale Satzlänge aus Kompatibilitätsgründen mit der K-Welt der o.a. logischen Blockgröße (und nicht der effektive Blocklänge) entspricht, kann die Länge eines Satzes die effektive Blockgröße übersteigen. In diesem Fall wird ein zu dem Datenblock gehöriger Überlaufblock angelegt. Der Teil eines Satzes, der nicht im Datenblock untergebracht werden kann, wird in einem 4 KB großen Überlaufblock abgelegt. Die Lage eines Schlüssels und seine Länge unterliegen den gleichen Beschränkungen wie beim NK2-ISAM-Format; d.h. er darf weder im NK4- noch im NK2-ISAM-Format ganz oder teilweise im Überlaufblock liegen.

Indexdatenblöcke bestehen aus mindestens einem und maximal acht 4-KB-Blöcken, wobei die Blockgröße der des Datenblocks entspricht.

Indexblöcke bestehen aus genau einem 4-KB-Block. Die Indexinformation ist zweistufig in Sections und Entries strukturiert, wobei die Länge einer Section der neuen Indexblockgröße von 4 KB angepasst ist.

Der Kontrollblock ist der erste Block einer ISAM-Datei. Er besteht aus genau einem 4-KB-Block. Er hat das gleiche Layout wie im NK2-Format. Der 2 KB übersteigende Platz wird nicht verwendet.

Freie Blöcke, die aus Daten- oder Indexdatenblöcken entstanden sind, bestehen aus mindestens einer und höchstens acht 4-KB-Blöcken. Freie Blöcke, die aus Index- oder Überlaufblöcken entstanden sind, bestehen aus genau einem 4-KB-Block.

Eine NK4-ISAM-Datei muss mit einem geradzahligen Blockungsfaktor angelegt werden. Sie kann folgendermaßen erstellt werden:

- automatisch, falls die Datei auf einer NK4-Platte (NK4-PVS) liegt

- wenn der Benutzer dies explizit angegeben hat (Kommando ADD-FILE-LINK ...,BLOCK-CONTROL-INFO=WITHIN-DATA-4K-BLOCK; Makro FILE/FCB ...,BLKCTRL=DATA4K)

Zu dem Thema Pufferung von NK4-ISAM-Dateien in ISAM-Pools siehe Abschnitt „NK4-ISAM-Dateien in ISAM-Pools “.

21.1.7 Primärindexblock (ISAM-Indexblock) und Sekundärindexblock

Primärindexblock

Ausgehend von den im Abschnitt „ISAM-Datenblock“ beschriebenen Datenblöcken übernimmt das DVS für jeden Datenblock den jeweils höchsten Schlüssel, verknüpft mit der logischen Blocknummer des Datenblocks, in einen Primärindexblock. Bei NK-ISAM wird aus diesem Schlüssel ein „Separator“ gebildet, bei K-ISAM werden ggf. auch Wertmarkierung und logische Markierung in den Indexeintrag übernommen.

Jeder Primärindexblock entspricht genau einer PAM-Seite. Reicht ein Primärindexblock nicht aus für die Verweise auf alle Datenblöcke, wird nicht nur ein zweiter Primärindexblock auf der gleichen Ebene angelegt, sondern auch ein Primärindexblock auf einer zweiten Ebene, dessen Einträge auf die Primärindexblöcke der ersten Ebene verweisen. Auf diese Weise entsteht ein Primärindexbaum.

Primärindexbaum

Die Baumstruktur einer ISAM-Datei wird gebildet durch die Primärindex- und Datenblöcke. Die Datenblöcke bilden die unterste Ebene (Ebene 0). Die übrigen Ebenen bestehen nur aus Primärindexblöcken. Die höchste Ebene besteht nur aus einem einzigen Primärindexblock, der auch als „Wurzel“ des Primärindexbaums bezeichnet wird und den Einstiegspunkt bildet für alle Zugriffe auf Sätze einer ISAM-Datei.

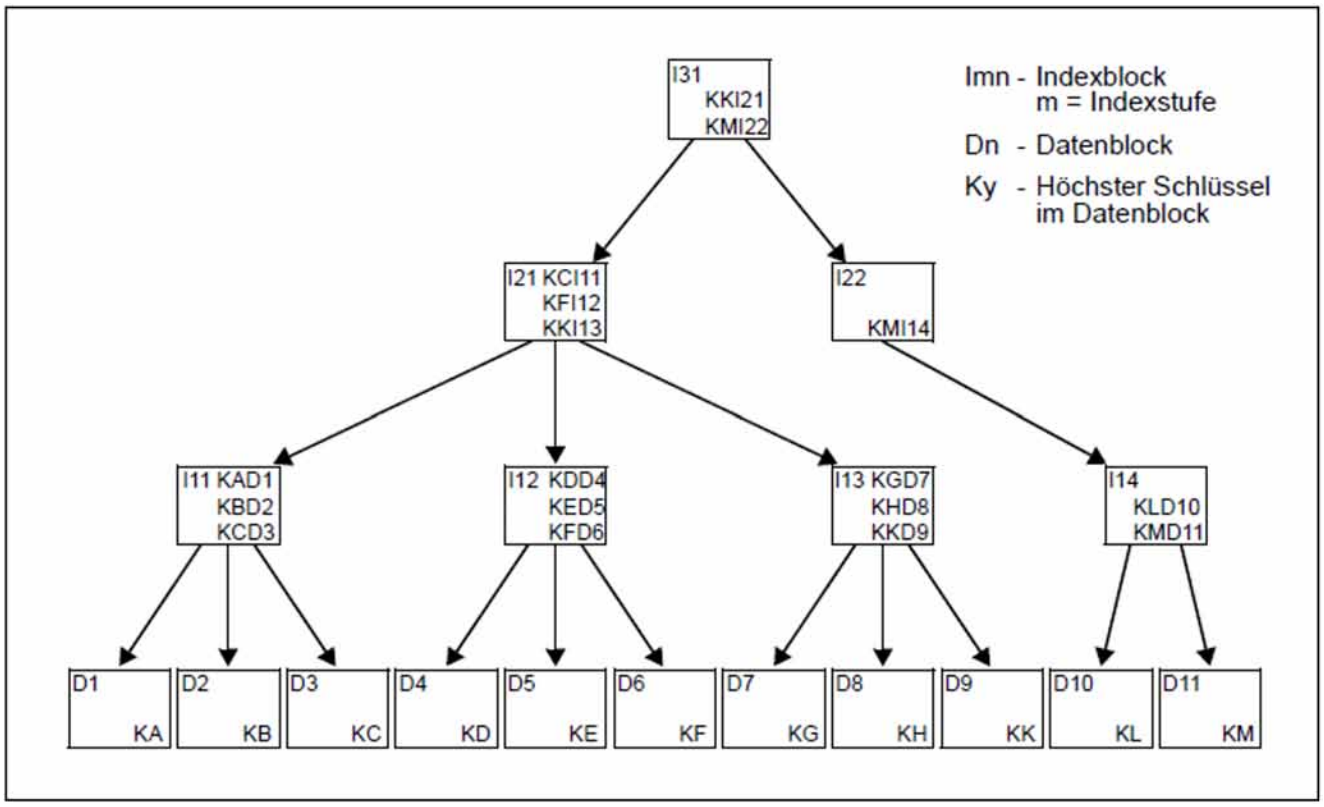


Bild 32: Primärindexbaum

Die im Beispiel gezeigte K-ISAM-Datei hat drei Primärindexebenen (I1, I2, I3) und die Datenebene (D). Die Wurzel des Primärindexbaums ist hier der Primärindexblock I31. Die Buchstaben Ky stellen den jeweils höchsten Schlüssel im Datenblock dar. Diese Schlüssel werden mit einem Verweis auf den zugehörigen Datenblock in die Primärindexblöcke der Ebene I1 aufgenommen. Für jeden Primärindexblock I1n wird wieder der höchste vorkommende Schlüssel, diesmal mit Verweis auf den Primärindexblock, in einen Primärindexblock der Ebene I2 aufgenommen. Da auch in dieser zweiten Ebene ein Primärindexblock nicht ausreicht, um alle Einträge aufzunehmen, muss eine dritte Primärindexebene erzeugt werden (I3), die nur noch aus einem Primärindexblock besteht, der Wurzel.

Sekundärindexblock (NK-ISAM)

Zu jedem in der Datei definierten Sekundärschlüssel legt das DVS einen Sekundärindex aus einem oder mehreren Sekundärindexblöcken an. In ihm hinterlegt es für jeden in der Datei vorkommenden Wert dieses Sekundärschlüssels einen Eintrag, der den Primärschlüsselwert des zugehörigen Datensatzes als Verweis enthält. Einträge mit identischem Sekundärschlüsselwert werden zu einem Satz zusammmengefasst. Würde ein solcher Satz die Grenzen eines Sekundärindexblockes überschreiten, wird ein weiterer Satz zum gleichen Sekundärschlüsselwert gebildet.

Ein Satz in einem Sekundärindex besteht aus

- dem Sekundärschlüsselwert mit dem größten Zeitstempel, der im Satz enthalten ist. Statt des Zeitstempels ist der Wert X'FF...FF' enthalten, wenn es sich um den letzten Satz mit dem Sekundärschlüsselwert handelt.
- einem oder mehreren Einträgen: Jeder Eintrag besteht aus dem Primärschlüsselwert des betreffenden Datensatzes und einem Zeitstempel. Wenn für den Sekundärschlüssel mehrfach auftretende Werte nicht zugelassen sind (Angabe DUPKEY=NO im Makro CREAIX bzw. DUPLICATE-KEY=*NO im Kommando CREATE-ALTERNATE-INDEX), enthält jeder Satz nur einen Verweis und dessen Zeitstempel den höchstmöglichen Wert (X'FF...FF').

Für einen Satz im Sekundärindexblock zu einem bestimmten Sekundärschlüsselwert ergibt sich daraus folgender Aufbau:

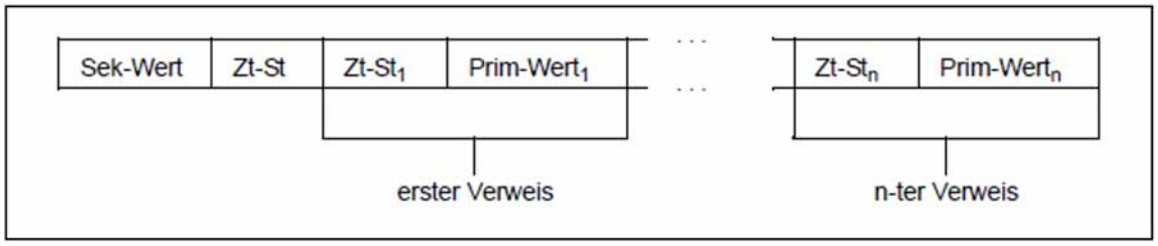


Bild 33: Aufbau eines Satzes im Sekundärindexblock

Im Bild bedeutet:

Sek-Wert	Sekundärschlüsselwert, für den dieser Satz angelegt wurde
Zt-St	Zeitstempel des Satzes
$Zt\text{-}St_i$	i=1,...,n: Zeitstempel des i-ten Eintrags
$Prim\text{-}Wert_i$	i=1,...,n: Primärschlüsselwert des Datensatzes, auf den der i-te Verweis zeigt

Zugriff auf Sätze in einer ISAM-Datei

Zugriff über den Primärschlüssel (ISAM-Schlüssel)

Wird eine ISAM-Datei nichtsequenziell verarbeitet, muss das DVS dem Benutzerprogramm den gewünschten Datensatz direkt zur Verfügung stellen können. Es beginnt mit der Suche nach einem Satz mit dem gewünschten Schlüssel immer in der obersten Primärindexebene – bei der Wurzel des Primärindexbaums – und wird dann (abwärts) durch alle Primärindexebenen bis zum benötigten Datenblock geführt. Die Suche innerhalb eines Primärindexblocks wird bei NK-ISAM dadurch beschleunigt, dass Primärindexeinträge zu „Sections“ zusammengefasst werden, sodass die Suche in Sprüngen verläuft.

Im Beispiel Bild 32 könnte z.B. der Satz mit dem Schlüssel KI gesucht werden. Aus den Primärindexeinträgen in I31 ist ersichtlich, dass die Suche im Primärindexblock I21 fortgesetzt werden muss: der Schlüssel KI ist kleiner als KK.
Die Einträge im Primärindexblock I21 verweisen für KI auf den Primärindexblock I13: KF < KI < KK.
Der Primärindexblock I13 enthält schließlich den Verweis auf den Datenblock D9: KH < KI < KK.
Der gesuchte Satz ist jetzt schnell zu finden.

Sowohl bei NK-ISAM als auch bei K-ISAM sind die Zugriffspfade über eine Baumstruktur realisiert, bei K-ISAM enthält der Primärindexblock der höchsten Stufe allerdings noch einen 36 Bytes langen Header mit Verwaltungsinformationen, bei NK-ISAM sind die Verwaltungsinformationen im Kontrollblock zu finden.

Durch die Schlüsselkomprimierung in den Primärindexblöcken ist bei NK-ISAM die Länge eines Primärindexeintrags nahezu unabhängig von der Schlüssellänge. Damit hängt die Zahl der Primärindexeinträge und -stufen nur noch von der Zahl der Datenblöcke ab.

Bei K-ISAM sind die Anzahl der Primärindexblöcke und der Primärindexebenen einer ISAM-Datei nicht nur abhängig von der Zahl der Datenblöcke, sondern auch von der Länge des ISAM-Schlüssels, bzw. von der Länge des ISAM-Primärindex, wenn Wert- oder logische Markierung genützt werden: es erfolgt keine Schlüsselkomprimierung, bei einer Primärindexlänge von 255 Bytes enthält ein Primärindexblock also nur 4 Indexeinträge.

Zugriff über einen Sekundärschlüssel (NK-ISAM)

Wenn Sie einen Datensatz über den Wert eines Sekundärschlüssels anfordern, so sucht das DVS zunächst in den zugehörigen Sekundärindexblöcken nach einem Satz mit diesem Wert. Ist ein solcher Satz vorhanden, so ermittelt es aus den darin enthaltenen Einträgen den Primärschlüsselwert des gewünschten Datensatzes. Bei mehreren Einträgen im Satz (mehrfach auftretende Sekundärschlüsselwerte) wählt es stets den Ersten aus. Mit diesem Primärschlüsselwert sucht das DVS dann den angeforderten Datensatz wie im vorhergehenden Abschnitt beschrieben.

Die Suche eines Datensatzes über einen Sekundärschlüssel ist demnach aufwändiger als seine Bereitstellung über den Primärschlüssel: Es werden etwa doppelt so viele Ein-/Ausgaben und die doppelte CPU-Zeit benötigt.

21.1.8 Freie Blöcke

Freie Blöcke enthalten keine Daten. Sie sind jedoch bereits von ISAM verwendet worden und werden von ISAM verwaltet. Sie entstehen beim Blocksplitting oder wenn beim Streichen von Sätzen aus der Datei Blöcke leer werden.

> **i** Freie Blöcke dürfen nicht mit „ungenutzten Blöcken“ verwechselt werden: sie gehören zum „genutzten Bereich“ einer ISAM-Datei; ihr Speicherplatz kann nicht per FILE (Operand SPACE) bzw. MODIFY-FILE-ATTRIBUTES freigegeben werden; „ungenutzte Blöcke“ bilden den „ungenutzten Bereich“ der ISAM-Datei: ihr Speicherplatz ist frei verfügbar.

Werden neue Daten-, Überlauf- oder Indexblöcke benötigt, werden bevorzugt freie Blöcke wieder verwendet.

21.1.9 Kontrollblock (NK-ISAM)

Der erste Block einer NK-ISAM-Datei (LBN=1) ist immer der Kontrollblock. Er enthält Verwaltungsinformationen, z. B. über die Dateigröße (File Size), den genutzten Bereich, die freien Blöcke usw. Der Kontrollblock darf nicht mit der „Wurzel“ des Indexbaums verwechselt werden, dem obersten Indexblock, der den Einstiegspunkt für die Suche über den Satzschlüssel bildet.

21.1.10 Größenberechnung von ISAM-Dateien

Die Berechnung der Dateigröße lässt sich für NK-ISAM und K-ISAM einheitlich durchführen. Dazu wird zunächst die Anzahl der Datenblöcke berechnet aus Satzlänge, nutzbarer Blockgröße und Anzahl der Datensätze.

nutzbare Blocklänge:

```
RECSIZE <= (n * 2048 (100 - PAD) / 100)
    ==> nutzbare Blocklänge = n * 2048 (100 - PAD) / 100
(n * 2048 (100 - PAD) / 100) <= RECSIZE <= n * 2048
    ==> nutzbare Blocklänge = RECSIZE
```

n	Blockungsfaktor
PAD	PAD-Wert (siehe PAD-Operand im Makro FILE)
RECSIZE	Satzlänge

Größe des Datenteils:

```
Anzahl Sätze pro Datenblock = (nutzbare Blockgröße / RECSIZE)
Anzahl Datenblöcke = (Sätze pro Datei / Sätze pro Datenblock)
```

Für NK-ISAM ist zusätzlich die Anzahl der Überlaufblöcke abzuschätzen. Sie ist abhängig von der Satzlänge:

Überlaufblöcke:

```
Anzahl Überlaufblöcke = i * Anzahl Datenblöcke
```

i = 0	für RECSIZE <= n * 2032 - 16
i = 1	für RECSIZE > n * 2032 - 16

Primärindexblöcke sequenziell erstellter NK-ISAM-Dateien enthalten im Durchschnitt ca. 200 Primärindexeinträge. Damit ergibt sich folgende Abschätzung:

Dateigröße:

```
Anzahl PAM-Seiten = Anzahl Datenblöcke *(BLKSIZE + i+ 0.005)
```

i = 1 / i= 0 berücksichtigt Überlaufblöcke (siehe oben)

Bei NK-ISAM-Dateien kann die Gesamtzahl der Primärindexblöcke einer Datei näherungsweise mit der Zahl der Primärindexblöcke in der untersten Primärindexebene gleichgesetzt werden. Aus den folgenden Tabellen lässt sich die Größe von NK-ISAM-Dateien leicht ableiten.

Bei sequenziell erstellten NK-ISAM-Dateien enthält jeder Primärindexblock ca. 200 Primärindexeinträge (Schlüsselkomprimierung!); daraus lässt sich folgende Tabelle ableiten:

Anzahl Datenblöcke	Anzahl Primärindexebenen
0 – 200	1
200 – 40000	2
40000 – 8000000	3

Tabelle 50: Berechnung der Primärindexebenen für sequenziell erstellte NK-ISAM-Dateien

Für nicht-sequenziell erstellte Dateien wächst die Anzahl der Primärindexebenen auf Grund des Blocksplittings schneller als für sequenziell erstellte Dateien. Im Mittel enthält ein Primärindexblock ca. 160 Primärindexeinträge, wenn mehr als eine Primärindexebene existiert.

Anzahl Datenblöcke	Anzahl Primärindexebenen
0 – 200	1
200 – 32000	2
32000 – 5120000	3

Tabelle 51: Berechnung der Primärindexebenen für nicht-sequenziell erstellte NK-ISAM-Dateien

Bei einer Datei mit Sekundärschlüsseln kann die Anzahl der Sekundärindexblöcke gemäß der nachfolgenden Formel abgeschätzt werden. In dieser Formel ist berücksichtigt, dass versucht wird, die Sekundärindexblöcke zumindest zu 75 % zu füllen. Dieser Füllgrad stellt nur einen Mittelwert dar. Es kann Fälle geben, in denen diese Zahl nicht erreicht werden kann.

Sekundärindexblöcke für den i-ten Sekundärschlüssel:

$$\text{Anzahl PAM-Seiten} = \text{Anzahl Datensätze} * (\text{P-KEYLEN} + 8 + h_i * (\text{S-KEYLEN}_i + 8)) / 1512$$

PAM-Seiten_i	PAM-Seiten der Sekundärindexblöcke für den i-ten Sekundärschlüssel
P-KEYLEN	Länge des Primärschlüssels der Datei
h_i	Relative Häufigkeit, mit der im i-ten Sekundärschlüssel der Datei verschiedene Werte vorkommen: Haben zum Beispiel in der Datei durchschnittlich jeweils 10 Datensätze den gleichen Wert im i-ten Sekundärschlüssel, so ist $h_i = 0{,}1$.
S-KEYLEN_i	Länge des i-ten Sekundärschlüssels

Die Gesamtzahl der Sekundärindexblöcke für alle Sekundärschlüssel in der Datei ergibt sich dann als Summe der Beiträge aller vereinbarten Sekundärschlüssel:

Sekundärindexblöcke einer Datei mit n Sekundärschlüsseln

$$\text{Anzahl PAM-Seiten} = \text{Anzahl PAM-Seiten}_1 + \ldots + \text{Anzahl PAM-Seiten}_n$$

K-ISAM

Für K-ISAM-Dateien ist die Anzahl der Primärindexeinträge von der Schlüssellänge abhängig:

$$\text{Anzahl der Einträge pro Primärindexblock} = 1024 / (\text{KEYLEN} + 4)$$

In dieser Formel ist berücksichtigt, dass für Primärindexblöcke nur ein Füllgrad von 50 % erreichbar ist. Die Gesamtgröße einer K-ISAM-Datei errechnet sich dann folgendermaßen:

Dateigröße:

$$\text{Anzahl PAM-Seiten} = \text{Anzahl Datenblöcke} * (\text{BLKSIZE} + ((\text{KEYLEN} + 4) / 1024))$$

KEYLEN: Schlüssellänge

21.2 ISAM-Pools

Vor allem die nicht-sequenzielle Verarbeitung von ISAM-Dateien hat oft eine hohe Ein-/Ausgaberate zur Folge. NK-ISAM reduziert diese Ein-/Ausgaberate und verbessert die Performance erheblich, indem es die Dateien in geeignet dimensionierten ISAM-Pools verarbeitet, die als Zwischenspeicher zur Pufferung von Dateiblöcken dienen.

> **i** K-ISAM verwendet solche Pools nicht. Mit „ISAM“ ist in der folgenden Beschreibung also immer nur NK-ISAM und nie K-ISAM gemeint.

ISAM-Pools können sowohl explizit vom Benutzer über spezielle Kommandos oder Makroaufrufe angelegt und verwaltet werden als auch implizit vom DVS:

- Vom DVS implizit eröffnete Pools heißen **Standard-ISAM-Pools** (siehe Abschnitt„Standard-ISAM-Pools“).
- Vom Benutzer erstellte Pools heißen **Benutzer-ISAM-Pools** (siehe Abschnitt „Benutzer-ISAM-Pools“).

Des Weiteren werden zwei Geltungsbereiche für Pools unterschieden:

- Ein **tasklokaler** ISAM-Pool (SCOPE=*TASK) wird im Klasse-5-Speicher der Eigentümertask abgelegt. Andere Tasks können auf diesen Speicher - und damit den ISAM-Pool - nicht zugreifen.
  Ein tasklokaler ISAM-Pool ist daher nur für ISAM-Dateien geeignet, die im Modus SHARUPD=NO eröffnet werden.
- Ein **taskübergreifender** ISAM-Pool (SCOPE=*HOST-SYSTEM) wird in einem privilegierten Data Space abgelegt, auf den alle Tasks im System zugreifen können. Näheres zu diesem verbesserten Konzept finden Sie im Abschnitt „ISAM-Pools in Data Spaces“.
  Ein taskübergreifender Pool ist Voraussetzung, um ISAM-Dateien im Modus SHARUPD=YES verarbeiten zu können. Im Modus SHARUPD=NO eröffnete ISAM-Dateien könnten zwar ebenfalls in Task-übergreifenden Pools bearbeitet werden. Aus Performance-Gründen (z.B. überflüssige Serialisierung) ist dies jedoch nicht zu empfehlen.

Die Geltungsbereiche SCOPE=*USER-GROUP und SCOPE=*USER-ID werden nur aus Kompatibilitätsgründen akzeptiert. Intern werden sie jedoch auf SCOPE=*HOST-SYSTEM (taskübergreifender Pool) abgebildet.

Die Kommandos und Makros zur Verwaltung von tasklokalen ISAM-Pools sind auch RFA zugänglich (siehe Handbuch „RFA“ [6]).

Eine Übersicht über die ISAM-Pool-Makroaufrufe ist im Abschnitt „Benutzer-ISAM-Pools“ zu finden.

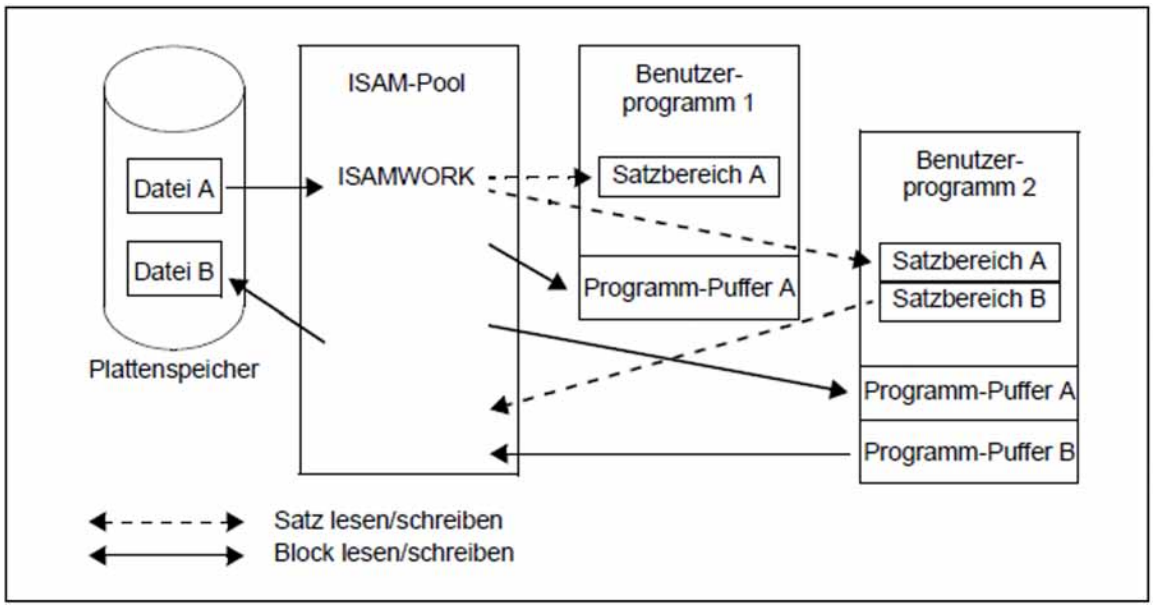


Bild 34: ISAM-Pool

Die Datei A wird von den Benutzerprogrammen PROG A und PROG B gelesen, die Datei B von PROG B erstellt. Der ISAM-Pool ISAMWORK ist ein taskübergreifender Pool.

21.2.1 Funktionen von ISAM-Pools

ISAM-Pools dienen zur Pufferung einer oder mehrerer NK-ISAM-Dateien. Neben den für die einzelnen PAM-Seiten benötigten Puffern enthalten sie Verwaltungsdaten, die u.a. die in ihnen gepufferten Dateien, die an sie angeschlossenen Aufträge und Zugriffe auf die einzelnen Puffer beschreiben.

Wird im Rahmen einer ISAM-Verarbeitung ein Block einer ISAM-Datei benötigt, wird zunächst der ISAM-Pool, in dem die Datei gepuffert wird, nach diesem Block durchsucht. Nur wenn der Block im Pool nicht gefunden wird, muss er von der Platte eingelesen werden. Sind alle Puffer des Pools besetzt, wird einer der bereits im Puffer befindlichen Blöcke überschrieben, ausgewählt nach folgenden Kriterien: Datenblöcke und Überlaufblöcke werden bevorzugt vor Indexblöcken ersetzt; innerhalb eines Blocktyps wird der Block ersetzt, auf den am längsten nicht mehr zugegriffen wurde.

Die Behandlung geänderter Blöcke ist für ISAM-Pools (und auch für einzelne Dateien) über Operanden einstellbar. Abhängig von den gewählten Verarbeitungseigenschaften werden geänderte Blöcke sofort oder erst bei Bedarf, d. h. wenn der Pufferinhalt überschrieben werden soll, auf Platte geschrieben (sofortiges Zurückschreiben: siehe "WROUT-Funktion").

Die folgende Tabelle enthält eine Übersicht über Makros und Kommandos zur Bearbeitung von tasklokalen ISAM-Pools und zur Anforderung von Informationen über diese.

Makro	Kommando	Kurzbeschreibung
ADDPLNK	ADD-ISAM-POOL-LINK	Zuordnen eines Poolkettungsnamens zu einem ISAM-Pool (taskspezifisch).
CREPOOL	CREATE-ISAM-POOL	Erzeugen eines neuen ISAM-Pools oder herstellen einer Verbindung der aufrufenden Task zu einem bereits existierenden ISAM-Pool.
DELPOOL	DELETE-ISAM-POOL	Aufheben der Verbindung der aufrufenden Task zu einem ISAM-Pool. Ist der Aufrufer die letzte bzw. einzige Task, die Verbindung zu einem ISAM-Pool hat, so wird auch der ISAM-Pool selbst entfernt. Es können auch alle Verbindungen der aufrufenden Task zu ISAM-Pools entfernt werden.
REMPLNK	REMOVE-ISAM-POOL-LINK	Aufheben der Zuordnung eines Pool-Kettungsnamens zu einem ISAM-Pool. Auch hier gibt es (wie bei DELPOOL/DELETE-ISAM-POOL) die Möglichkeit, durch Spezifikation eines speziellen Parameters mit einem Aufruf alle (taskspezifischen) Zuordnungen von Pool-Kettungsnamen zu ISAM-Pools zu entfernen.
SHOPOOL	SHOW-ISAM-POOL-ATTRIBUTES	Gibt Informationen über einen ISAM-Pool aus, zu dem der Aufrufer zum Zeitpunkt des Aufrufes einen Anschluss besitzt. Es ist auch möglich, Informationen über sämtliche ISAM-Pools zu erhalten, an die die aufrufendeTask derzeit angeschlossen ist.
SHOPLNK	SHOW-ISAM-POOL-LINK	Gibt die Zuordnung von ISAM-Pools zu ISAM-Pool-Kettungsnamen aus.

Tabelle 52: ISAM-Pool-Makros und -Kommandos

21.2.2 Standard-ISAM-Pools

Wird eine ISAM-Datei eröffnet, ohne dass sie vom Benutzer im Makro FILE/FCB (Operand POOLLNK) oder im Kommando ADD-FILE-LINK (Operand POOL-LINK) einem bestimmten ISAM-Pool zugeordnet wurde, verwendet das DVS für die Dateiverarbeitung einen Standard-ISAM-Pool des Systems: Bei einem OPEN mit SHARUPD=NO wird ein tasklokaler Standard-ISAM-Pool zugeordnet, bei einem OPEN mit SHARUPD=YES ein taskübergreifender Standard-ISAM-Pool.

Ein **tasklokaler** ISAM-Pool ist dadurch gekennzeichnet, dass er nur von einer Task genutzt werden kann. Beim ersten Eröffnen einer ISAM-Datei mit SHARUPD=NO legt das DVS den ISAM-Pool $TASK01 an. Dieser Pool wird auch für weitere ISAM-Dateiverarbeitungen des Auftrags genutzt, solange für jede Datei ausreichend Platz zur Verfügung steht. Andernfalls wird beim Eröffnen der nächsten ISAM-Datei der Pool $TASK02 erzeugt usw.

Maximal stehen pro Auftrag 16 ISAM-Pools zur Verfügung ($TASKn, 01 <= n <= 16). Die Größe der Pools legt die Systembetreuung über den ISAM-Parameter LCLDFPS fest.

Ein **taskübergreifender** ISAM-Pool kann von allen Tasks im System genutzt werden. Beim ersten Eröffnen einer ISAM-Datei mit SHARUPD=YES legt das DVS einen solchen Pool an und verwendet ihn in der Regel auch nur für diese Datei. Für weitere mit SHARUPD=YES eröffnete Dateien werden, sofern der verfügbare Platz ausreicht, jeweils eigene Pools angelegt.

Im Gegensatz zu tasklokalen Pools haben taskübergreifende Pools keinen Namen. Sie sind implizit an die enthaltene Datei geknüpft. Ihre Größe legt die Systembetreuung über den ISAM-Parameter GLBPS fest, ihre maximale Anzahl indirekt über den ISAM-Parameter MAXDSBN. Dieser Parameter, der im laufenden Betrieb mit dem Kommando MODIFY-ISAM-CACHING geändert werden kann, gibt an, wie viele Data Spaces das DVS für die Aufnahme von taskübergreifenden ISAM-Pools verwenden darf. Aus der festen Maximalgröße eines Data Spaces (2 GB) und der Größe der Pools ergibt sich die Anzahl der Pools, die maximal erzeugt werden kann.

> ¡ Das Konzept der taskübergreifenden Standard-ISAM-Pools $SYS01 bis $SYS16, die in Memory Pools abgelegt werden, existiert seit BS2000/OSD-BC V6.0B nicht mehr.

Ein Auftrag kann gleichzeitig mehrere ISAM-Dateien verarbeiten, die verschiedenen Standardpools zugeordnet sein können.

Beim Schließen einer ISAM-Datei, die mit einem Standardpool verarbeitet wurde, prüft das DVS zunächst, ob der Auftrag noch über weitere Dateien mit diesem Pool verbunden ist. Wenn das nicht der Fall ist, gibt es die von diesem Pool belegten Ressourcen frei.

Die Seiten eines taskübergreifenden ISAM-Pools in einem Data Space, die Benutzerdaten enthalten, werden nicht in Speicherauszüge ausgegeben. In evtl. zu erstellenden Speicherauszügen erscheinen von solchen Pools lediglich die zur Diagnose erforderlichen Verwal-tungs-Daten.

21.2.3 Benutzer-ISAM-Pools

Sie können **tasklokale** ISAM-Pools über Makroaufrufe oder Kommandos erzeugen und verwalten.

Der Pool wird allerdings nur dann für die Dateiverarbeitung genutzt, wenn ihm mit dem Makro ADDPLNK bzw. dem Kommando ADD-POOL-LINK ein Poolkettungsname zugeordnet und mit dem Operanden POOLLNK bzw. POOL-LINK eine Verbindung zwischen ISAM-Pool und Datei hergestellt wird.

Poolkettungsname – Verbindung zwischen Datei und ISAM-Pool

Mit dem Makro ADDPLNK bzw. dem Kommando ADD-POOL-LINK wird einem ISAM-Pool in einem Auftrag ein Poolkettungsname zugeordnet und in die Pooltabelle des Auftrags eingetragen. Dieser Poolkettungsname muss dann für jede Datei, die in diesem Pool verarbeitet werden soll, mit dem Operanden POOLLNK bzw. POOL-LINK in die TFT eingetragen werden. Außerdem muss mit BLKCTRL=DATA bzw. mit BLOCK-CONTROL-INFO= WITHIN-DATA-BLOCK die NK-ISAM-Verarbeitung eingestellt werden.

Beim Eröffnen einer Datei prüft das DVS, ob in FCB oder TFT ein Poolkettungsname eingetragen ist und ob zu diesem Poolkettungsnamen ein ISAM-Pool existiert (über die Pooltabelle). Ist dies der Fall und sind Pool- und Dateieigenschaften verträglich, wird die Datei über diesen ISAM-Pool verarbeitet.

Anmerkungen zu taskübergreifenden Benutzer-ISAM-Pools

Benutzer-ISAM-Pools sind stets **tasklokal**.

Kommandos und Makros für taskübergreifende Benutzer-ISAM-Pools werden aus Kompatibilitätsgründen weiter akzeptiert, führen jedoch nicht mehr zur Erzeugung eines solchen Pools. Stattdessen verwendet das DVS einen Standard-ISAM-Pool, wobei die vom Benutzer zuvor angegebene Poolgröße nicht unterschritten und die Zuordnung von Dateien zu Pools beachtet wird.

Das folgende Diagramm zeigt, welche Aktionen von einem Auftrag ausgelöst werden müssen, der an der Programmschnittstelle mit einem Benutzer-ISAM-Pool arbeitet.

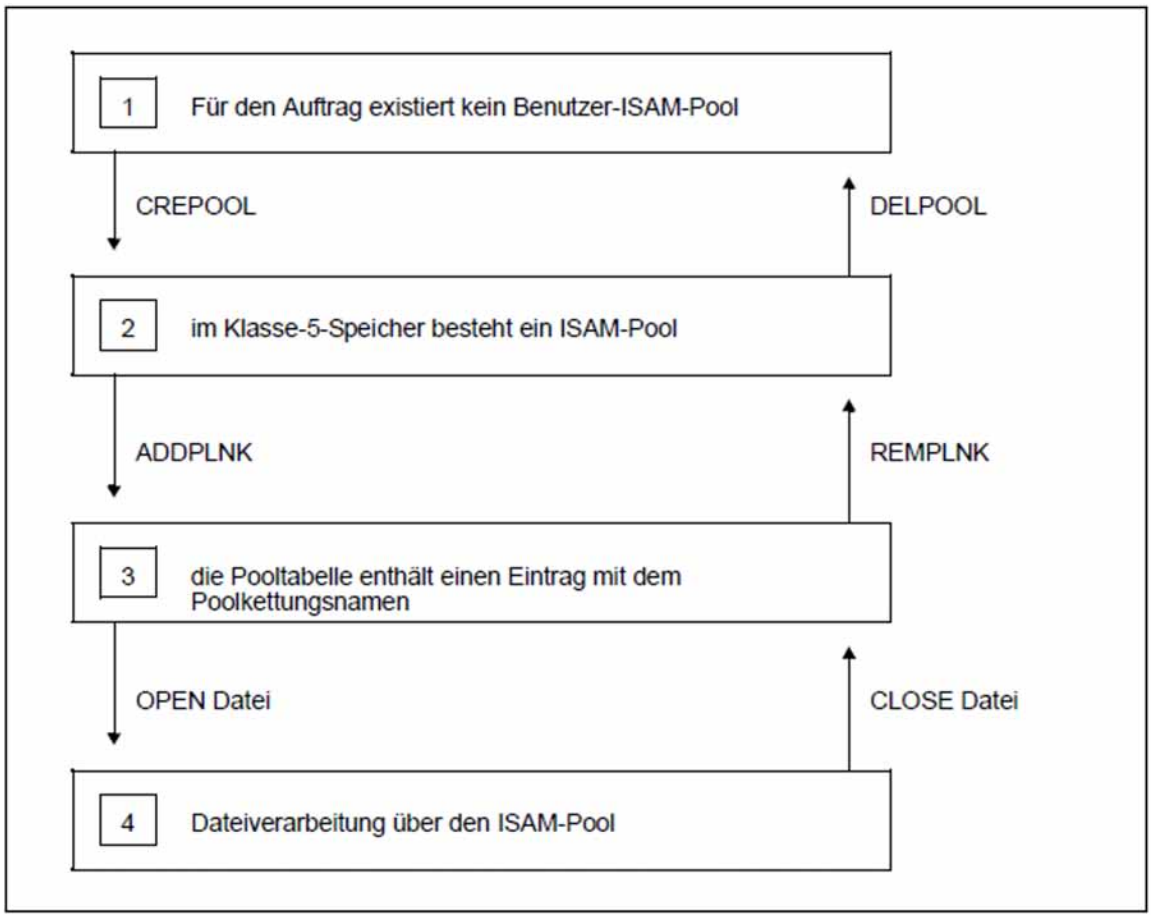


Bild 35: Aktionen zur Dateiverarbeitung mit ISAM-Pools (Makros)

Die folgende Tabelle zeigt, welche Makroaufrufe dem Benutzer zur Verfügung stehen.

Makro	**Funktion**	**Aktion vgl. Bild 35**
CREPOOL	ISAM-Pool erzeugen bei taskübergreifenden Pools wird geprüft, ob der Pool bereits existiert	1 -> 2
ADDPLNK	Poolkettungsname definieren der Poolkettungsname wird in die Pooltabelle eingetragen	2 -> 3
FILE/FCB LINK=name POOLLNK=name BLKCTRL=DATA	beim OPEN Verbindung zwischen ISAM-Pool und Datei herstellen	3 -> 4 (OPEN) 4 -> 3 (CLOSE)
REMPLNK	Eintrag des Poolkettungsnamens in der Pooltabelle löschen	3 -> 2
DELPOOL	ISAM-Pool „löschen“: Speicherplatz im Klasse-5-Speicher freigeben / Verbindung zum taskübergreifenden Pool aufheben,wenn noch andere Aufträge den Pool nutzen	2 -> 1

Tabelle 53: Übersicht über die ISAM-Pool-Makroaufrufe

Das folgende Diagramm zeigt, welche Aktionen von einem Auftrag ausgelöst werden müssen, der an der Kommandoschnittstelle mit einem Benutzer-ISAM-Pool arbeitet.

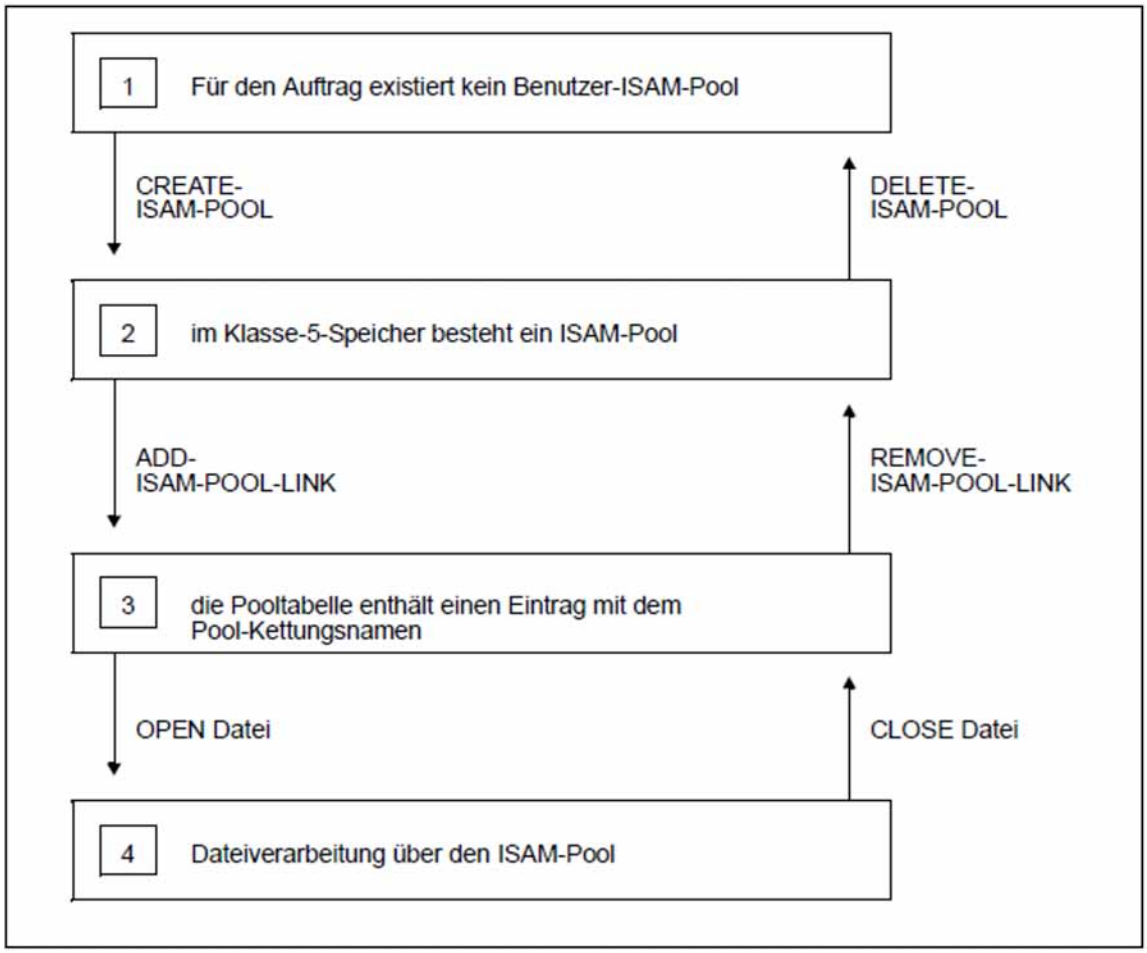


Bild 36: Aktionen zur Dateiverarbeitung mit ISAM-Pools (Kommandos)

Die folgende Tabelle zeigt, welche Kommandos dem Benutzer zur Verfügung stehen.

Kommando	**Funktion**	**Aktion vgl. Bild 36**
CREATE-ISAM-POOL	ISAM-Pool erzeugen bei taskübergreifenden Pools wird geprüft, ob der Pool bereits existiert	1 -> 2
ADD-ISAM-POOL-LINK	Poolkettungsname definieren der Poolkettungsname wird in die Pooltabelle eingetragen	2 -> 3
ADD-FILE-LINK LINK-NAME=name POOL-LINK=name BLOCK-CONTR-INFO= *WITHIN-DATA-BLOCK	beim OPEN Verbindung zwischen ISAM-Pool und Datei herstellen	3 -> 4 (OPEN) 4 -> 3 (CLOSE)

REMOVE-ISAM-POOL-LINK	Eintrag des Poolkettungsnamens in der Pooltabelle löschen	3 -> 2
DELETE-ISAM-POOL	ISAM-Pool „löschen“: Speicherplatz im Klasse-5-Speicher freigeben / Verbindung zum taskübergreifenden Pool aufheben, wenn noch andere Aufträge den Pool nutzen	2 -> 1

Tabelle 54: Übersicht über die ISAM-Pool-Kommandos

21.2.4 NK4-ISAM-Dateien in ISAM-Pools

Für die Pufferung von NK4-Dateien werden ISAM-Pools nicht wie bisher in Einheiten von 2-KB-Blöcken, sondern auch in Einheiten von 4-KB-Blöcken angelegt. Werden in einem ISAM-Pool sowohl NK2- als auch NK4-Dateien gepuffert, so besteht der ISAM-Pool aus zwei Extents, einem Extent zu Aufnahme von 2-KB-Blöcken und einem Extent zur Aufnahme von 4-KB-Blöcken. Dabei wird der erste Extent beim Anlegen des ISAM-Pools unformatiert bereitgestellt und bei Eröffnung der ersten Datei, die über ihn gepuffert wird, in entsprechenden Einheiten (2 KB oder 4 KB) angelegt. Ein zweiter Extent wird angelegt, wenn eine Datei mit einem anderen Blockkontrollattribut über diesen ISAM-Pool eröffnet werden soll. Dieser Extent erhält ebenfalls die für Blocklänge angegebene Größe.

NK2-Dateien und NK4-Dateien sollten nicht über den gleichen ISAM-Pool gepuffert werden, da sich bei Optimierung der Poolgröße der Platzbedarf verdoppeln kann (es wird jeweils ein 2-KB- und ein 4-KB-Extent in der angegebenen Größe angelegt).

21.2.5 Größenberechnung von ISAM-Pools

Die Größe eines ISAM-Pools wirkt sich auf die Performance der Dateiverarbeitung aus. Für die Optimierung /Anpassung der Poolgröße an die Dateiverarbeitung muss Folgendes beachtet werden:

- Wie viele Dateien sollen über den ISAM-Pool verarbeitet werden?
- Wie sollen sie verarbeitet werden: sequenziell oder nichtsequenziell?
- Wie groß sind Index- und Datenbereich bzw. die Datenblöcke?
- Wie viele Aufträge können auf den Pool zugreifen?

Im Folgenden wird zunächst die Größenberechnung eines taskspezifischen ISAM-Pools für eine Datei erklärt. Soll der Pool mehrere Dateien aufnehmen können, ist diese Berechnung für jede Datei durchzuführen und die Ergebnisse sind zu addieren. Soll der ISAM-Pool auch für weitere Aufträge zugänglich sein, muss der Platzbedarf für jeden Auftrag berechnet und die Ergebnisse addiert werden. Einen Sonderfall bilden Dateien, die gleichzeitig von mehreren Aufträgen bearbeitet werden (SHARUPD=YES): der Platzbedarf für den Indexbereich braucht nur einmal berücksichtigt zu werden.

Zur Größenberechnung von Index- und Datenteilen von ISAM-Dateien siehe Abschnitt„Größenberechnung von ISAM-Dateien“.

Sequenzielle Dateiverarbeitung

Bei rein sequenzieller Dateiverarbeitung wird die Performance von der Poolgröße kaum beeinflusst: bearbeitete Blöcke werden nicht noch einmal benötigt, sodass der ISAM-Pool sehr klein gehalten werden kann. Er sollte gleichzeitig aufnehmen können: den Kontrollblock, die benötigten Indexblöcke und zwei Datenblöcke inklusive evtl. nötiger Überlaufblöcke. Für die Poolgröße gilt – näherungsweise – folgende Formel:

Poolgröße:

```
SIZE = 1.1 * (1 + i + 2m)
```

i = Anzahl der Indexstufen der Datei
m = Anzahl der PAM-Seiten pro Datenblock inklusive Überlaufblock (m = n+1, n aus BLKSIZE=(STD,n))

Der Faktor 1.1 berücksichtigt, dass ca. 10% des Pool-Platzbedarfs für Verwaltungsdaten genutzt werden.

Die Gesamtzahl der Indexblöcke einer ISAM-Datei kann näherungsweise mit der Zahl der Indexblöcke in der untersten Indexebene gleichgesetzt werden. Für die Berechnung der optimalen Poolgröße bedeutet dies: für nichtsequenziell verarbeitete ISAM-Dateien wird zunächst der Platzbedarf für den Indexbereich ermittelt; da auch einige Datenblöcke im Pool Platz haben sollten, muss dieser Wert nach oben gerundet werden.

Anzahl Indexblöcke der Ebene 1:

```
Anzahl = LASTPG / (n * 160)
```

LASTPG = Anzahl der PAM-Seiten der Datei
n = Blockungsfaktor (BLKSIZE=(STD,n))

Der Wert 160 entspricht der durchschnittlichen Anzahl Indexeinträge pro Indexblock.

Beispiel: Poolgröße

Für die Datei ISAM.B gilt: LASTPG = 18270; ISAM.B wurde mit BLKSIZE=(STD,2) definiert und enthält keine Überlaufblöcke.

```
LASTPG / n = 18270 / 2 = 9135
```

Die Datei enthält ca. 9135 Datenblöcke.

```
9135 / 160 > 57
```

Die unterste Indexebene besteht aus 57 Indexblöcken. Die optimale Poolgröße läge für diese Datei zwischen 60 und 70 PAM-Seiten: außer dem gesamten Indexbereich könnte der Pool mehrere Datenblöcke aufnehmen.

Kleine Dateien

Kleine Dateien können u.U. ganz in den ISAM-Pool aufgenommen werden.

Poolgröße für kleine ISAM-Dateien:

```
SIZE  1.1 * LASTPG
```

LASTPG ist die höchste von ISAM benutzte PAM-Seite und zeigt damit die Dateigröße an.

Indexblöcke im ISAM-Pool

Günstig ist, wenn alle Indexblöcke in den ISAM-Pool aufgenommen werden können. Zusätzlich sollte noch für jeden auf die Datei zugreifenden Auftrag Platz für einen Datenblock vorgesehen werden. Der Platzbedarf lässt sich folgendermaßen errechnen.

Platzbedarf:

```
SIZE >= 1.1 * (LASTPG / (160 * BLKSIZE) + #TASK * BLKSIZE)
```

„#TASK“ ist die Anzahl gleichzeitig auf die Datei zugreifender Aufträge; BLKSIZE der Blockungsfaktor.

Minimale Poolgröße

minimale Poolgröße:

$$SIZE_{MIN} = 1{,}1 * ((\#INDEX - 1 + BLKSIZE) * \#TASK + 1)$$

Außer Blockgröße und Anzahl „paralleler Aufträge“ muss die Anzahl der Indexebenen (#INDEX) einer Datei berücksichtigt werden. #INDEX kann folgendermaßen abgeschätzt werden:

Anzahl Indexebenen:

$$\#INDEX = [\ \ln (LASTPG / BLKSIZE) / \ln(160)\ ]$$

„ln“ ist der natürliche Logarithmus, „[„ und „]“ bedeuten, dass der Gesamtausdruck auf die nächsthöhere ganze Zahl zu runden ist.

Poolspezifische Grenzwerte

Die Anzahl Dateien (#FILE) und die Anzahl gleichzeitig zugreifender Aufträge (#TASK) für einen Pool sind durch die SIZE-Angabe begrenzt. Es muss gelten:

Grenzwerte:

```
    #FILE + #TASK <= SIZE * 1.4
```

Wirksam ist nur der intern durch Rundung entstehende SIZE-Wert, nicht der vom Benutzer angegebene.

Beispiel

Im Makro CREPOOL bzw. dem Kommando CREATE-ISAM-POOL definierte Poolgröße: SIZE = 32 . Die Summe der eröffneten Dateien und der angeschlossenen Aufträge darf den Wert 44 nicht überschreiten.

21.2.6 ISAM-Pools in Data Spaces

Das Anlegen von Pufferbereichen für NK-ISAM-Dateien (ISAM-Pools), die mit SHARUPD=YES geöffnet werden, erfolgt automatisiert und optimiert. Somit kann die explizite Konfiguration von ISAM-Pools über die entsprechenden Programm- oder Kommandoschnittstellen entfallen.

Das Anlegen von ISAM-Pools erfolgt dateispezifisch beim ersten Öffnen einer Datei. Im Rahmen der Dateieröffnung wird ein passender Abschnitt eines Data Space als ISAM-Pool initialisiert. Dabei wird automatisch die Größe bestimmt:
Der Data Space mit einer Gesamtgröße von max. 2 GB ist in gleich große Abschnitte zu je 1 MB (oder einem 2^n-fachen davon) eingeteilt.

Ein ISAM-Pool besteht mindestens aus einem derartigen Abschnitt.
Abhängig von der Dateigröße kann ein ISAM-Pool auch aus mehreren benachbarten Abschnitten dieses Data Space bestehen. Die Pool-Größe wird auf Basis der Formel auf "Größenberechnung von ISAM-Pools" geschätzt.

Weiter besteht die Möglichkeit, mit dem Kommando ADD-FILE-LINK ...,POOL-SIZE= oder über die entsprechende Programmschnittstelle FILE explizit eine Größe zu vereinbaren. Diese Angabe wird jedoch nur für ISAM-Pools in Data Spaces, also für Dateibearbeitung mit SHARUPD=YES, ausgewertet.

> **i** Kommandos und Makros für taskübergreifende Benutzer-ISAM-Pools werden aus Kompatibilitätsgründen weiter akzeptiert, führen jedoch nicht mehr zur Erzeugung eines solchen Pools. Stattdessen verwendet das DVS einen Standard-ISAM-Pool, wobei die vom Benutzer zuvor angegebene Poolgröße nicht unterschritten und die Zuordnung von Dateien zu Pools beachtet wird.

Der Systemverwalter kann sich mit Hilfe des Kommandos SHOW-ISAM-CACHING über in privilegierten Data Spaces gepufferte ISAM-Dateien informieren.

21.3 Dateiverarbeitung

Datei- und Verarbeitungseigenschaften werden im FCB, im Makro FILE bzw. im Kommando ADD-FILE-LINK definiert.

21.3.1 Dateieigenschaften

> ¡ Um die Lesbarkeit zu wahren, nehmen die folgenden Kurzbeschreibungen ausschließlich Bezug auf die Operanden von FILE- und FCB-Makro. Die adäquaten Operanden des Kommandos werden am Ende dieses Abschnitts in einer Tabelle dargestellt.

Zugriffsmethode und Dateiformat

Die Zugriffsmethode wird mit dem Operanden FCBTYPE festgelegt, ISAM ist Standardwert; in Abhängigkeit vom Operanden BLKCTRL wird die Datei mit NK-ISAM oder mit K-ISAM verarbeitet.

Das Dateiformat (BLKCTRL=DATA/PAMKEY/NO) bezieht sich auf den internen Aufbau der Datei: für NK-ISAM werden die Blockkontrollinformationen in der PAM-Seite selbst geführt (BLKCTRL=DATA), nicht in einem gesonderten PAM-Schlüssel. K-ISAM nützt diesen PAM-Schlüssel: BLKCTRL=PAMKEY. Das Dateiformat BLKCTRL=NO (kein Blockkontrollfeld) entspricht der Angabe BLKCTRL=DATA.

Aufbau einer PAM-Seite

K-ISAM: die Blockkontrollinformationen stehen in einem der PAM-Seite vorangestellten PAM-Schlüssel; die ersten 16 Byte der PAM-Seite enthalten Benutzerdaten.

NK-ISAM: jede PAM-Seite beginnt mit einem 16 Byte langen Blockkontrollfeld, dessen letzte 4 Bytes ISAM-spezifische Informationen enthalten.
Die nutzbare „Länge“ einer PAM-Seite ist demnach 2048 – 16 = 2032 Bytes.

NK2-Dateien: Jede PAM-Seite eines logischen Blocks beginnt mit einem 16 Byte langen Blockkontrollfeld, dessen letzte 4 Byte ISAM-spezifische Informationen enthalten.
Die nutzbare Länge einer PAM-Seite ist 2048-16=2032 Byte.

NK4-Dateien: Jede physikalische Seite (4096 Byte) beginnt mit einem 16 Byte langen Blockkontrollfeld, dessen letzte 4 Byte ISAM-spezifische Informationen enthalten.
Die nutzbare Länge einer solchen Seite ist 4096-16=4080.

Satz- und Blockformate

Mit ISAM können Sätze fester und Sätze variabler Länge verarbeitet werden, definiert über den Operanden RECFORM=F/V. Das Satzformat wirkt sich aus auf die möglichen Definitionen von Satzlänge und Schlüsselposition (siehe Abschnitt „ISAM-Dateistrukturen“).

Die Satzlänge wird definiert mit dem Operanden RECSIZE. Bei Sätzen vom Format V ist zu berücksichtigen, dass die ersten 4 Bytes des Satzes für Satzlängenfeld und ein Steuerfeld genutzt werden. Außerdem gibt RECSIZE dann nicht die aktuelle Satzlänge, sondern die maximale Satzlänge an. Gibt es in der Datei Sätze, die länger sind, als in RECSIZE angegeben, wird bei Leseoperationen nur die RECSIZE entsprechende Anzahl Bytes eingelesen.

Bei Sätzen vom Format F steht die gesamte in RECSIZE angegebene Satzlänge für die Benutzerdaten zur Verfügung. Bei der RECSIZE-Definition muss jedoch berücksichtigt werden, dass ISAM intern auch diesen Sätzen ein 4-Byte-Satzlängen- und Steuerfeld voranstellt, was sich auf die Bildung von Überlaufblöcken auswirken kann.

Die maximal zulässige Satzlänge ist immer die Datenblocklänge (BLKSIZE).

Durch den Blockungsfaktor wird im Operanden BLKSIZE festgelegt, aus wie vielen PAM-Seiten die Datenblöcke bestehen sollen:

BLKSIZE=(STD,n), 1 <= n <= 16.

Die Blockgröße sollte immer – zusammen mit der Satzlänge – so gewählt werden, dass keine Überlaufblöcke entstehen, da dies zu Performanceverlusten führt.

Index-Aufbau

Der ISAM-Index wird gebildet aus ISAM-Schlüssel und Markierungen (Flags), er kann an beliebiger Stelle im Datensatz liegen, darf jedoch nicht in einen Überlaufblock hineinragen. Die maximale Indexlänge ist 255 Bytes, die minimale Schlüssellänge ein Byte. Markierungen werden mithilfe des GETFL-Makroaufrufs ausgewertet.

Der ISAM-Index beginnt immer mit dem Schlüssel, die Anfangsposition wird definiert durch den Operanden KEYPOS. Die Voreinstellungen für KEYPOS berücksichtigen das Satzformat: KEYPOS=5 für RECFORM=V, KEYPOS=1 für RECFORM=F. Die Schlüssellänge wird über KEYLEN definiert, standardmäßig erzeugt das DVS 8 Byte lange Schlüssel.

Für die Flags sind die Operanden VALLEN und LOGLEN relevant, der Operand VALPROP hat bei NK-ISAM keine Funktion mehr, er wird noch aus Kompatibilitätsgründen unterstützt, da er bei K-ISAM die Aufnahme der Wertmarkierung in den Indexeintrag steuert.

Die folgende Tabelle stellt die oben genannten Operanden der Makros FILE und FCB den Operanden des Kommandos ADD-FILE-LINK gegenüber.

Operand in den Makros FILE und FCB	Operand im Kommando ADD-FILE-LINK
FCBTYPE	ACCESS-METHOD
BLKCTRL=DATA/PAMKEY/NO	BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK / *PAMKEY / *NO
RECFORM=F/V	RECORD-FORMAT=*FIXED / *VARIABLE
RECSIZE	RECORD-SIZE
BLKSIZE	BUFFER-LENGTH
KEYPOS	KEY-POSITION
VALLEN	VALUE-FLAG-LENGTH
LOGLEN	LOGICAL-FLAG-LENGTH
VALPROP	PROPAGATE-VALUE-FLAG

Tabelle 55: ISAM-Operanden in FILE/FCB und ADD-FILE-LINK

21.3.2 Verarbeitungseigenschaften

Im Folgenden werden die Verarbeitungseigenschaften von ISAM-Dateien beschrieben; dazu zählen auch Shared-Update-Verarbeitung und Datei-Eröffnung, die gesondert in AAbschnitt „Shared-Update-Verarbeitung“ und Abschnitt „ISAM-Datei eröffnen“ beschrieben werden.

Datei- und Poolkettungsname

Der Dateikettungsname verknüpft über die TFT (Task File Table) Datei und Dateisteuerblock im Programm (siehe Abschnitt „Zugriff über den Dateikettungsnamen (Link-Name)“).

Soll eine NK-ISAM-Datei in Benutzer-ISAM-Pools verarbeitet werden, muss im Makro ADDPLNK mit dem Operanden POOLLNK bzw. im Kommando ADD-POOL-LINK mit dem Operanden POOL-LINK ein Poolkettungsnamen angegeben werden, der eine Verbindung zu einem ISAM-Pool herstellt. Dieser Pool kann mit dem Makro CREPOOL bzw. dem Kommando CREATE-ISAM-POOL erzeugt werden. Mit dem Makro ADDPLNK bzw. dem Kommando ADD-POOL-LINK wird dann der Poolkettungsname festgelegt. Wird weder im FILE- noch im FCB-Makroaufruf POOLLNK angegeben bzw. der Operand POOL-LINK im Kommando ADD-FILE-LINK nicht angegeben, werden NK-ISAM-Dateien in Standardpools verarbeitet. K-ISAM-Dateien können nicht in ISAM-Pools verarbeitet werden.

Mehrfachschlüssel

Sie können in den Makros FILE und FCB (Operand DUPEKY) bzw. im Kommando ADD-FILE-LINK (Operand DUPLICATE-KEY) festlegen, dass verschiedene Datensätze einer Datei identische Schlüssel enthalten können. Bei NK-ISAM versieht das DVS intern diese Sätze mit einem Zeitstempel, bei K-ISAM gibt es keinen zusätzlichen Zeitstempel.

Blockfüllung: PAD-Wert

Bei sequenzieller Dateierstellung mit PUT können Sie über den Operanden PAD in den Makros FILE und FCB bzw. über den Operanden PADDING-FACTOR im Kommando ADD-FILE-LINK bestimmen, wie viel Platz in den Datenblöcken freibleiben soll für spätere Aktualisierung. Standardmäßig gilt PAD=15, d.h. 15 % des Speicherplatzes im Datenblock bleiben frei. Wenn mit einem PUT-Aufruf diese Grenze erreicht wird, wird für den folgenden Satz ein neuer Datenblock angelegt.

Für NK-ISAM-Dateien gilt: sobald diese PAD-Grenze überschritten ist, wird ein neuer Datenblock angefordert; K-ISAM fordert einen neuen Datenblock bereits an, bevor die PAD-Grenze überschritten wird.

Beim sequenziellen Erstellen (PUT) nimmt der Platzbedarf einer Datei mit steigendem PAD-Faktor zu. Durch geeignete Wahl des PAD-Faktors lässt sich jedoch die nachfolgende Dateiverarbeitung (STORE/INSRT) optimieren: in den Datenblöcken ist so viel freier Platz

vorzusehen, dass es bei Dateierweiterungen nicht zum Blocksplitting kommt. Zur richtigen Wahl des PAD-Faktors ist also eine Prognose erforderlich, in welchem Umfang die Datei wachsen wird.

Werden ISAM-Dateien mit STORE erstellt, hat PAD keinen Einfluss auf die Blockfüllung: STORE schreibt solange Sätze in einen Datenblock, bis dieser gefüllt ist. Wird noch ein weiterer Satz geschrieben, der nicht mehr in den Datenblock passt, kommt es zum Blocksplitting; die Blöcke werden in der Regel nur zu 50 % gefüllt.

Auch wenn die Datei zwar mit PUT, aber nicht über einen Ein-/Ausgabebereich im Programm erstellt wird, ist die PAD-Angabe wirkungslos (der Ein-/Ausgabebereich wird im FCB mit IOAREAn definiert; siehe unter „Programmpuffer = Ein-/Ausgabebereich im Benutzerprogramm“). Jeder PUT-Aufruf löst eine Schreiboperation aus, und das DVS versucht, den aktuellen Satz im letzten Datenblock unterzubringen. Erst wenn dieser gefüllt ist, legt es einen neuen Datenblock an.

Überlappendes Lesen

Soll eine ISAM-Datei vorwiegend sequenziell verarbeitet werden, können Sie mit dem Operanden OVERLAP=YES im Makro FILE bzw. mit dem Operanden READ-IN-ADVANCE im Kommando ADD-FILE-LINK festlegen, dass bei jedem Leseaufruf „prophylaktisch“ auch der benachbarte Datenblock in den ISAM-Pool eingelesen wird. Auf diese Weise kann die Ein-/Ausgaberate reduziert werden. Überlappendes Lesen erfolgt sowohl beim „Vorwärts-Lesen“ (in Richtung Dateiende) als auch beim „Rückwärts-Lesen“ (in Richtung Dateianfang).

WROUT-Funktion

Die WROUT-Funktion (bzw. Write-Immediate-Funktion) steuert das Zurückschreiben geänderter Blöcke auf die Platte. Bei eingeschalteter WROUT-Funktion ist die Konsistenz der Datei auf Platte und im virtuellen Speicher zu jedem Zeitpunkt gewährleistet.

Ist WROUT nicht eingeschaltet, erfolgt ein Zurückschreiben erst dann, wenn der Inhalt des betreffenden Pufferbereichs ersetzt werden muss. Durch dieses verzögerte Zurückschreiben lassen sich Schreiboperationen auf Platte einsparen, wenn im gleichen Block mehrfach geändert wird.

In nicht taskspezifischen ISAM-Pools ist WROUT=YES voreingestellt: geänderte Blöcke werden immer sofort zurückgeschrieben. Der Benutzer kann jedoch auch mit verzögertem Zurückschreiben auf Platte arbeiten. Dazu ist sowohl im Makro CREPOOL bzw. im Kommando CREATE-ISAM-POOL als auch im Makro FILE bzw. im Kommando ADD-FILE-LINK verzögertes Zurückschreiben auf Platte zu spezifizieren.

Wird die Datei in einem taskspezifischen Benutzer-ISAM-Pool verarbeitet, kann die WROUT-Funktion bei Verwendung von Makros sowohl in FILE/FCB für die Datei als auch im CREPOOL-Makro für den Pool eingeschaltet werden. Bei Verwendung von Kommandos kann die Funktion WRITE-IMMEDIATE sowohl im Kommando ADD-FILE-LINK für die Datei als auch im Kommando CREATE-ISAM-POOL für den Pool eingeschaltet werden. Bei unterschiedlichen Angaben auf Datei- und Pool-Ebene überwiegt jeweils die Angabe, bei der die WROUT-Funktion eingeschaltet wird.

Programmpuffer = Ein-/Ausgabebereich im Benutzerprogramm

Verwendet ein Programm einen eigenen Ein-/Ausgabebereich, muss dieser mindestens die Größe eines Datenblocks haben (= n * 2048 BLKSIZE, 1 <= n <= 16) entsprechend BLKSIZE=(STD,n). Standardmäßig wird vom System ein Ein-/Ausgabebereich im Klasse-5-Speicher angelegt.

Die Existenz eines vom Benutzer in seinem Programm mit IOAREA1/2 definierten Ein/Ausgabebereichs ist vor allem bei sequenzieller Verarbeitung von ISAM-Dateien von Vorteil durch die Reduzierung von SVCs:

- sequenzielles Lesen (GET/GETR): bei der ersten Leseoperation werden so viele Sätze wie möglich in den Ein-/Ausgabebereich übertragen, bevor der erste Satz dem Programm zur Verfügung gestellt wird. Bei den nachfolgenden Leseoperation werden dem Programm dann die weiteren im Ein-/Ausgabebereich vorhandenen Sätze zur Verfügung gestellt. Eine erneute Ein-/Ausgabe erfolgt erst, wenn alle Sätze gelesen sind.
- sequenzielles Schreiben (PUT): bei sequenzieller Dateierstellung oder -erweiterung werden die Sätze im Ein-/Ausgabebereich gesammelt, bis er gefüllt ist (PAD-Wert wird berücksichtigt) oder die sequenzielle Verarbeitung durch Aufruf einer anderen Operation beendet wird. Es ist also darauf zu achten, dass „PUT“-Folgen nicht durch andere Aktionsaufrufe unterbrochen werden, da dann jedes Mal aus dem Inhalt des Ein-/Ausgabebereichs ein Datenblock gebildet wird und so Datenblöcke mit geringer Blockfüllung entstehen können.

Für NK-ISAM kann im Übertragungsbetrieb (Move Mode) auf den Ein-/Ausgabebereich verzichtet werden (im FCB: IOAREA1=NO). Jeder Aktionsmakroaufruf führt dann zu einem SVC.

Betriebsarten: Übertragungs-/Ortungsbetrieb

Normalerweise laufen ISAM-Aktionen im Übertragungsbetrieb (Move Mode) ab. Einige ISAM-Aktionen können auch im Ortungsbetrieb (Locate Mode) ablaufen, der über den Operanden IOREG eingestellt wird. Da dann auch die Übertragung von Sätzen in einen oder von einem Ein-/Ausgabebereich entfällt, kann dies bei sequenzieller Dateiverarbeitung zu geringfügigen Performance-Verbesserungen führen. Der Ortungsbetrieb wird bei NK-ISAM nur noch aus Kompatibilitätsgründen unterstützt. Näheres siehe Kapitel „Zugriffsmethoden“.

21.3.3 Index-/Datentrennung

K-ISAM-Dateien können mit voneinander getrennten Index- und Datenteilen auf verschiedenen Privatplatten eingerichtet werden. Bei den Dateiteilen können unabhängig Geräte, Datenträger und Speicherplatz zugewiesen werden.

> ¡ Die nachfolgende Beschreibung bezieht sich auf die Operanden des Makros FILE. Für Benutzer, die mit den DVS-Kommandos arbeiten, werden die entsprechenden Operanden der Kommandos in einer Tabelle zusammengefasst.

Für den Indexteil gelten im FILE-Aufruf die Operanden DEVICE, VOLUME und SPACE, für den Datenteil die Operanden DDEVICE, DVOLUME und DSPACE. Geräte-, Datenträger- und Speicherplatzreservierung erfolgen jeweils nach dem gleichen Prinzip. Sie müssen allerdings berücksichtigen, dass sich Angaben bei SPACE immer auf VOLUME und DEVICE und Angaben bei DSPACE auf DVOLUME und DDEVICE beziehen. Für die Gerätetypen bei DDEVICE sind nur die unter DEVICE genannten Geräteypen zulässig, für DSPACE gibt es keinen Standardwert bei Dateierstellung.

Bezieht sich der FILE-Aufruf auf eine Datei, die noch keinen Speicherplatz besitzt, müssen DDEVICE, DVOLUME und DSPACE immer zusammen angegeben werden. Für Dateien, denen bereits Speicherplatz zugewiesen wurde, kann DSPACE auch ohne DDEVICE und DVOLUME angegeben werden. Speicherplatzfreigabe ist nur für die Gesamtdatei möglich, nicht getrennt für Index- und Datenteil. Wurde eine ISAM-Datei mit getrenntem Index- und Datenteil erstellt, ist es später nicht mehr möglich, Index- und Datenblöcke auf den gleichen Datenträger zu bringen.

Eine solche Teilung einer ISAM-Datei ist auf gemeinschaftlichen Datenträgern nicht möglich. NK-ISAM weist die Operanden DDEVICE, DSPACE, DVOLUME aus Kompatibilitätsgründen zwar nicht zurück, ignoriert sie jedoch bei der Verarbeitung. Auf Grund einer optimierten Pufferverwaltung ist bei NK-ISAM eine Trennung der Datei in Index- und Datenteil nicht mehr erforderlich.

Nachfolgend die Tabelle mit den entsprechenden Operanden der Kommandos:

Operand im Makro	Operand in den Kommandos CREATE-FILE, MODIFY-FILE-ATTRIBUTES
DEVICE	DEVICE-TYPE
VOLUME	VOLUME
SPACE	SPACE
DDEVICE	DATA-DEVICE-TYPE
DVOLUME	DATA-VOLUME
DSPACE	DATA-SPACE

Tabelle 56: Operanden in Makro und Kommandos für Index-/Datentrennung

21.4 Shared-Update-Verarbeitung

ISAM-Dateien können von verschiedenen Aufträgen gleichzeitig eröffnet und aktualisiert werden. Jeder Auftrag muss im entsprechenden FILE- oder FCB-Aufruf bzw. im Kommando ADD-FILE-LINK die Shared-Update-Funktion einschalten (SHARUPD=YES bzw. SHARED-UPDATE=*YES). Die Datenkonsistenz wird von ISAM durch interne Sperren gewährleistet.
Sie können auch explizit – über den ISAM-Schlüssel – Datensätze sperren. Die Sperre wird in den Makroaufrufen GET, GETFL, GETKY, GETR mit der Angabe LOCK angefordert. Kann eine Sperre nicht gesetzt werden, weil der betreffende Satz/Bereich bereits von einem anderen Auftrag gesperrt ist, wird der Auftrag in eine Warteschlange eingereiht (es sei denn, im Programm ist eine Routine vorgesehen, die beim Auftreten von Page-Lock-Ereignissen (EXLST-Ausgang PGLOCK) aktiviert werden soll).

21.4.1 Sperren

Das Sperren von Datensätzen ist in NK-ISAM und K-ISAM unterschiedlich realisiert:in NK-ISAM als Schlüsselsperren, in K-ISAM als Blocksperren.

Schlüsselsperren (NK-ISAM)

Soll ein Satz in die Datei eingefügt (STORE, INSRT) oder mit Schlüsselangabe gelöscht (ELIM) werden, wird der entsprechende Satz von ISAM automatisch gesperrt und nach der Verarbeitung „entsperrt“. Auch wenn Sie die Sperre explizit gefordert haben (externe Sperre mit LOCK), gibt ISAM sie nach Abschluss der Schreiboperation frei – allerdings nicht bei sequenziellen Schreiboperationen (PUT), die über einen Ein-/Ausgabebereich erfolgen. Es wird ein Schlüsselbereich gesperrt, dessen untere Grenze kleiner oder gleich dem Schlüssel der Sätze im Puffer ist. Seine obere Grenze ist „high value“ (X'FF').

Soll ein Datensatz geändert werden, geht der Schreiboperation eine Leseoperation voraus. Sie müssen dann bereits beim Leseaufruf den Satz sperren (GET LOCK). Die Sperre wird automatisch aufgehoben, wenn der Satz zurückgeschrieben (PUTX) oder gelöscht ist (ELIM ohne Schlüsselangabe).

Leseoperationen (GET, GETFL, GETKY, GETR) können mit oder ohne Sperre ausgeführt werden. Nicht gesperrte Sätze können von weiteren Benutzern gelesen, aber auch verändert werden. Eine für eine Leseoperation angeforderte Sperre wird von ISAM nach Abschluss der Leseoperation nicht automatisch aufgehoben.
Auch wenn ein Satz über einen Sekundärschlüssel gelesen wird, bezieht sich die Sperre auf den Primärschlüssel des gelesenen Satzes.

Beim sequenziellen Lesen wird, wenn mit Ein-/Ausgabebereich gearbeitet wird, ein Schlüsselbereich gesperrt, dessen untere bzw. obere Grenze durch den kleinsten bzw. größten Schlüssel der im Puffer enthaltenen Sätze gegeben ist.

Sperren freigeben

Sie können explizit die für Ihren Auftrag bestehende Sperre in der Datei aufheben (ISREQ); dies kann z.B. erforderlich sein, wenn Sie einen Satz für eine Leseoperation gesperrt haben und der Leseoperation keine weitere ISAM-Aktion für die Datei folgt. Da jeder Auftrag nur eine Sperre besitzen darf, hebt jede ISAM-Operation, die explizit oder implizit eine Sperre anfordert, die vorher bestehende Satzsperre auf, sodass keine Deadlock-Situationen entstehen können.

Blocksperre (K-ISAM)

Bei K-ISAM sind alle Sperren als Blocksperren realisiert; d.h., der Datenblock, der den angesprochenen Satz enthält, ist für alle anderen Benutzer gesperrt.

21.4.2 OPEN-Modi bei Shared-Update-Verarbeitung

Der erste Benutzer, der eine ISAM-Datei eröffnen will, kann jede mögliche Kombination der bei OPEN und SHARUPD zulässigen Werte angeben. Die nachfolgende Tabelle gibt Aufschluss darüber, welche Kombination OPEN/SHARUPD für den Benutzer B zulässig ist, wenn Benutzer A die Datei bereits eröffnet hat. Wurde die Datei bereits von mehr als einem Benutzer eröffnet, wird die OPEN/SHARUPD-Kombination des Benutzers B mit allen übrigen OPEN/SHARUPD-Kombinationen verglichen. Der Benutzer B kann die Datei nur dann eröffnen, wenn die vorausgegangenen Vergleiche es zulassen.

			USER B									
	SHARUPD=		***YES**					***NO**				
		OPEN-Modus	INPUT	INOUT	EXTEND	OUTIN	OUTPUT	INPUT	INOUT	EXTEND	OUTIN	OUTPUT
USER A	***YES**	INPUT	X	X	X			X		X		
		INOUT	X	X						X		
		EXTEND	X	X								
		OUTIN	X	X								
		OUTPUT	X	X								
	***NO**	INPUT	X					X				
		INOUT										
		EXTEND										
		OUTIN										
		OUTPUT										

Tabelle 57: ISAM: erlaubte SHARUPD-/OPEN-Kombinationen

X bedeutet: OPEN-Modus für Benutzer B erlaubt.

Hinweis für Zugriffsmethode NK-ISAM

Bei Zugriffen auf Dateien im Modus SHARUPD=YES kann der Fall eintreten, dass eine Datei mit einer Dateigröße < 32 GB durch entsprechende Verarbeitung zu einer Datei >= 32 GB wird.

Hier werden zwei Fälle unterschieden:

- Aufrufer, die auf diese Situation vorbereitet sind
  (mit der Angabe LARGE_FILE=*ALLOWED beim Makro FCB
  bzw. EXCEED-32GB=*ALLOWED beim Kommando ADD-FILE-LINK)
- nicht vorbereitete Aufrufer
  (Angabe LARGE_FILE=*FORBIDDEN beim Makro FCB bzw.
  EXCEED-32GB=*FORBIDDEN beim Kommando ADD-FILE-LINK).

Bei jedem SVC-Einstieg wird eine Überprüfung der Größe der betroffenen NK-ISAM-Datei anhand der im File Table Entry verankerten Extentliste durchgeführt. Wird dabei eine Dateigröße über 32 GB ermittelt und hat der Aufrufer in seinem FCB das Attribut LARGE_FI-LE=*FORBIDDEN gesetzt, wird die Verarbeitung abgebrochen. NK-ISAM liefert in diesem Fall den Returncode.

```
X'00000A23' FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT.
```

(bzw. die entsprechende DMS-Meldung `DMS0A23`)

Die Zugriffsmethode K-ISAM (BLKCTRL=PAMKEY) ist von dieser Problematik nicht betroffen.

21.4.3 ISAM-Pools für Shared-Update-Verarbeitung (NK-ISAM)

Dateien, die shared-update eröffnet werden sollen, müssen einem nicht-taskspezifischen oder hostspezifischen Benutzer-ISAM-Pool zugeordnet werden, damit auch andere Benutzer auf diesen Pool und damit auf die Datei zugreifen können. Vor Beginn der Dateiverarbeitung müssen Sie mit dem Makro CREPOOL, Operand SCOPE=HOST, bzw. mit dem Kommando CREATE-ISAM-POOL, Operand SCOPE=HOST-SYSTEM, einen hostspezifischen Pool erzeugen bzw. Ihren Auftrag an einen hostspezifischen Pool anschließen.

Mit dem Makro ADDPLNK bzw. dem Kommando ADD-POOL-LINK müssen Sie diesem Pool einen Poolkettungsnamen zuweisen, auf den Sie sich in den Makros FILE und FCB mit dem Operanden POOLLNK bzw. im Kommando ADD-FILE-LINK mit dem Operanden POOL-LINK beziehen müssen. Außerdem müssen Sie in FILE /FCB mit BLKCTRL=DATA bzw. in ADD-FILE-LINK mit BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK die NK-ISAM-Verarbeitung einstellen. Nach Abschluss der Dateiverarbeitung müssen Sie Poolkettungsnamen und Pool löschen (Makros REMPLNK, DELPOOL oder Kommandos REMOVE-ISAM-POOL, DELETE-ISAM-POOL).

Die folgenden Bilder zeigen, welche Kombinationen von OPEN-Modus, SHARUPD und Pool-Geltungsbereich möglich sind.

Beispiel 1: Eingabedatei

Eine Datei wird als Eingabedatei genutzt: sie kann beliebig vielen task-spezifischen und maximal einem nicht-task-spezifischen ISAM-Pool zugeordnet werden.

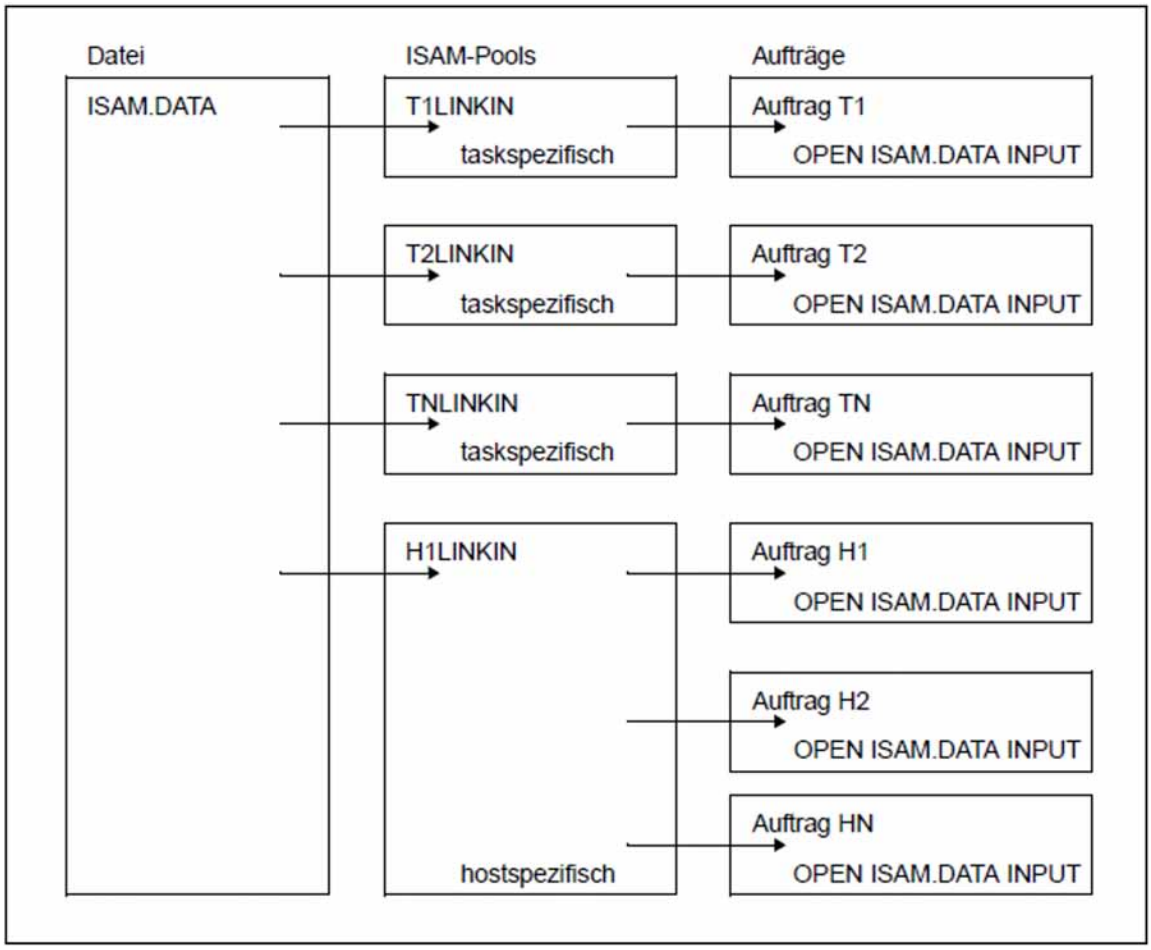


Bild 37: Eingabedatei in ISAM-Pools

Beispiel 2: Ausgabedatei

Eine Datei soll von mehreren Aufträgen gleichzeitig eröffnet werden und von mindestens einem Auftrag aktualisiert werden: sie muss über einen nicht-task-spezifischen ISAM-Pool verarbeitet und in allen beteiligten Aufträgen mit SHARUPD=YES eröffnet werden. Handelt es sich um einen Benutzer-ISAM-Pool, müssen alle Aufträge unter der gleichen Benutzerkennung ablaufen.

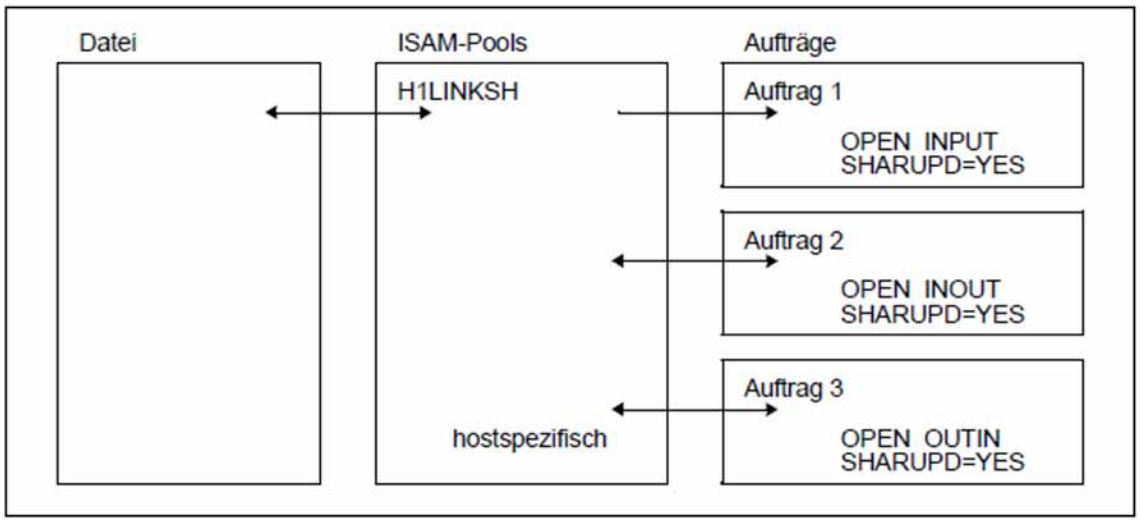


Bild 38: Shared-Update-Verarbeitung in ISAM-Pools

21.5 ISAM-Datei eröffnen

Bevor eine Datei verarbeitet werden kann, muss sie mit einem OPEN-Makroaufruf eröffnet werden. Für ISAM-Dateien sind folgende Eröffnungsmodi zulässig: OUTPUT, OUTIN, EXTEND, INOUT, INPUT. Gleichzeitig muss überprüft werden, ob die Datei bereits von einem anderen Auftrag geöffnet ist, in welchem ISAM-Pool sie verarbeitet werden soll, ob sie im Übertragungs- oder im Ortungsbetrieb zu verarbeiten ist, usw. In der folgenden Tabelle sind die OPEN-Modi zusammengestellt.

Open-Modus	Kurzbeschreibung
OUTPUT	Es wird eine neue Datei sequenziell erstellt, es ist nur der PUT-Makroaufruf erlaubt. Existiert bereits eine ISAM-Datei mit dem angegebenen Namen, wird sie überschrieben, der Katalogeintrag wird neu erstellt.
OUTIN	Wie bei OPEN OUTPUT wird eine neue Datei erstellt, eine evtl. vorher bestehende Datei wird überschrieben. Es sind alle ISAM-Aktionen zulässig.
EXTEND	Eine bestehende Datei wird sequenziell erweitert; wie bei OUTPUT sind nur Schreiboperationen mit PUT zulässig.
INOUT	Eine existierende Datei soll aktualisiert werden: wie bei OUTIN sind alle ISAM-Aktionenerlaubt wie Suchen, Lesen, Ändern, Einfügen und Löschen von Sätzen.
INPUT	Eine existierende Datei soll gelesen werden, d.h. es sind nur Leseoperationen zulässig.

Tabelle 58: Open-Modi bei Eröffnen einer ISAM-Datei

Dateisperre

Wird eine Datei nicht für Shared-Update-Verarbeitung geöffnet, hängt es vom OPEN-Modus ab, ob verschiedene Aufträge gleichzeitig mit der Datei arbeiten können. Wird eine Datei anders als INPUT eröffnet, ist sie für weitere Aufträge gesperrt; wird sie mit INPUT eröffnet, können andere Aufträge sie ebenfalls mit OPEN INPUT eröffnen.

Speicherplatzzuweisung

Die minimale Speicherplatzzuweisung muss berücksichtigen, dass außer dem Datenblock immer ein Indexblock entsteht, bei NK-ISAM noch zusätzlich der Kontrollblock. Das heißt: für BLKSIZE=(STD,n) muss die Primärzuweisung für NK-ISAM-Dateien mindestens n+2 PAM-Seiten betragen, für K-ISAM-Dateien n+1 PAM-Seiten. Da beim OPEN keine Sekundärzuweisung durchgeführt werden kann, kommt es zum OPEN-Fehler, wenn nicht ausreichend Speicherplatz zur Verfügung steht (zu Primärzuweisung siehe Abschnitt „Anfordernvon Speicherplatz“).

Betriebsarten

ISAM-Dateien werden normalerweise im Übertragungsbetrieb verarbeitet; Dateiverarbeitung im Ortungsbetrieb ist möglich, wird bei NK-ISAM allerdings nur noch aus Kompatibilitätsgründen unterstützt.

Aktionsmakro-Aufruf	OPEN-Typ				
	INPUT	OUTPUT	EXTEND	INOUT	OUTIN
GET	B	-	-	B	B
GETR	B	-	-	B	B

GETFL	B	-	-	B	B
GETKY	B	-	-	B	B
PUT	-	B	B	B	B
PUTX	-	-	-	B	B
INSRT	-	-	-	M	M
STORE	-	-	-	M	M
ELIM	-	-	-	x	x
SETL	x	-	-	x	x

Tabelle 59: ISAM-Aktionsmakroaufrufe und OPEN-Typ

B: Übertragungs- oder Ortungsbetrieb möglich

M: Übertragungsbetrieb (Ortungsbetrieb, nur wenn der Arbeitsbereich versorgt wurde)

x: Aktionsmakroaufruf zulässig

- Aktionsmakroaufruf nicht zulässig

21.5.1 Übernahme der Dateieigenschaften in den FCB

Die Inhalte von FCB-Feldern für Dateieigenschaften können vom Wert des entsprechenden Operanden im FILE-Aufruf abweichen; dies gilt z.B. für KEYPOS, KEYLEN, PAD und BLKSIZE. Wenn „n“ der Wert des Feldes KEYPOS und „m“ der Wert von KEYLEN im Katalogeintrag bzw. in TFT oder FCB-Makroaufruf ist, gilt für geöffnete Dateien:

- das Feld KEYLEN im P1-FCB hat den Inhalt m-1
- das Feld KEYPOS im P1-FCB hat für RECFORM = F den Inhalt (n+4)-1 = n+3

Der PAD-Wert wird nur dann berücksichtigt, wenn die Datei sequenziell mit PUT-Makroaufruf erstellt wird. Die Makroaufrufe INSRT und STORE verwenden den gesamten Speicherplatz eines Datenblocks, es kann jedoch schon bei Dateierstellung zum Blocksplitting kommen.

Auch der Inhalt des Feldes BLKSIZE im P1-FCB kann vom Wert des BLKSIZE-Operanden in FILE, FCB-Makroaufruf oder Katalogeintrag abweichen: außer bei INPUT-Dateien wird der BLKSIZE-Wert unter Berücksichtigung des PAD-Wertes neu berechnet (BLKSIZE minus PAD) und im FCB-Feld BLKSIZE hinterlegt.

Beispiel

> Katalogeintrag: BLKSIZE=(STD,3); PAD=15 (Standard).
> Der Inhalt von BLKSIZE im TU-FCB errechnet sich folgendermaßen: (3*2048) * (1.0-0.15); im Feld BLKSIZE steht während der Verarbeitung der Wert X'1467'.

21.5.2 DUMMY-Dateien

Für Testzwecke, vor allem für das Testen von Fehler-Routinen des Benutzerprogramms, eignen sich sog. DUMMY-Dateien, die im FILE-Makroaufruf mit *DUMMY bzw. im Kommando ADD-FILE-LINK mit *DUMMY definiert werden. Bei der Verarbeitung solcher DUMMY-Dateien finden keine Ein-/Ausgaben statt. Leseversuche führen zum Ansprung von Fehler-Routinen, Schreibaufrufe werden ignoriert, d.h. es wird in jedem Fall eine Nulloperation (d.h. keine Operation) durchgeführt.

Die folgende Tabelle zeigt, welche Ereignisse bei den ISAM-Leseoperationen auftreten.

Leseoperation	Makro	Ereignis	EXLST-Ausgang	Meldung
sequenzielles Lesen	GET/GETR	Dateiende	EOFADDR	DMS0AAE
Lesen mit Schlüssel	GETKY	Schlüssel nicht vorhanden	NOFIND	DMS0AA8
Lesen über Flags	GETFL LIMIT=KEY LIMIT=END	Schlüssel nicht vorhanden Dateiende	NOFIND EOFADDR	DMS0AA8 DMS0AAE

Tabelle 60: ISAM-Leseoperationen für DUMMY-Dateien

21.6 Einsatz von NK-ISAM

Einsatz von ISAM-Pools

Sie können entscheiden, ob Ihre Dateien in eigenen Benutzer-Pools verarbeitet werden sollen oder in Standardpools, die Ihnen das System zur Verfügung stellt.

Wird bei größeren Anwendungen Performance-Optimierung angestrebt, sollten unbedingt Benutzer-ISAM-Pools verwendet werden, da nur so die Poolgröße den Erfordernissen der Anwendung angepasst werden kann.

Standardpools können eingesetzt werden, wenn Sie nur gelegentlich und in geringem Umfang mit ISAM arbeiten. Auch am Beginn der Umstellung von K-ISAM auf NK-ISAM kann u.U. mit Standardpools gearbeitet werden.

Dimensionierung von ISAM-Pools

Für den Benutzer stellt sich die Frage nach der optimalen Poolgröße nur für die Benutzer-Pools. Die Möglichkeit, ganze Dateien in einen ISAM-Pool zu laden, sodass während der Dateiverarbeitung keine Ein-/Ausgaben durchzuführen sind, lässt sich nur für kleine Dateien realisieren: maximale Poolgröße ist 4096 PAM-Seiten.

ISAM-Pools sollten die Indexblöcke der Datei aufnehmen können und mindestens einen Datenblock pro Auftrag, der auf die Datei zugreift. Bei der Berechnung der optimalen Poolgröße muss auch der reale Speicher berücksichtigt werden, da große ISAM-Pools ohne ausreichenden Ausbau des Realspeichers eine Erhöhung der Paging-Rate zur Folge haben. Außerdem kann zwischen „aktiven Dateien“, die ständig im Zugriff sind, und Dateien, auf die nur gelegentlich zugegriffen wird, unterschieden werden: nur „aktive Dateien“ belegen Seiten des ISAM-Pools.

Sequenzielle Dateiverarbeitung in ISAM-Pools

Bei sequenzieller Verarbeitung lässt sich die Ein-/Ausgaberate nur geringfügig reduzieren. In taskspezifischen Pools erfolgt z.B. Blockwechsel durch lineares Fortschalten der Indexeinträge, sodass fast stets auf die gleichen Indexblöcke zugegriffen wird. Eine andere Möglichkeit zur Optimierung bieten große Blockungsfaktoren; bei rein sequenzieller Verarbeitung rentieren sich also keine großen Pools.

Satzlänge

Da jede PAM-Seite mit einem 16-Byte-Kontrollfeld beginnt, reduziert sich entsprechend der in einem Block zur Verfügung stehende Platz. Sollen keine Überlaufblöcke entstehen, gilt für die Satzlänge folgende Regel:

```
RECSIZE <= (n * 2048) - (n * 16) - 16 - F - T
```

F	Satzformat F = 0/4 für RECFORM = V/F
T	Zeitstempel T = 0/8 für Dateien ohne/mit Mehrfachschlüssel

(n * 2048) beschreibt die Blocklänge (BLKSIZE), (n * 16) beschreibt das Blockkontrollfeld am Beginn einer jeden PAM-Seite.

Flag-Verarbeitung

Da die Markierungen nicht mehr in den Index übernommen werden, ist bei NK-ISAM die Flag-Verarbeitung nur noch als sequenzielles Durchsuchen von Dateiabschnitten realisiert, was die Performance stark senken kann. Im GETFL-Makroaufruf sollte der zu durchsuchende Bereich über den LIMIT-Operanden möglichst klein gehalten werden.

Schlüsselposition und Indexlänge

Der ISAM-Index darf nicht in Überlaufblöcke hineinragen; es gilt also für den Indexaufbau Folgendes (Markierungen müssen – wenn vorhanden – berücksichtigt werden!):

```
KEYPOS + KEYLEN + VALLEN + LOGLEN <= n * 2032 - 16 - F
```

n	Blockungsfaktor in BLKSIZE=(STD,n), 1 <= n <= 16
F	Satzformat • F = 0 für RECFORM = V • F = 4 für RECFORM = F

ISAM-Pools und BLKCTRL = DATA ohne NK-ISAM

Ist NK-ISAM an einem Rechner nicht geladen und versucht ein Benutzer ISAM-Pools zu erzeugen oder NK-ISAM-Dateien (BLKCTRL=DATA) zu verarbeiten, wird die Verarbeitung mit einer Fehlermeldung abgebrochen. Für ISAM-Pool-Makros wird im Standardheader der Operandenliste ein Returncode hinterlegt. Der FCB der entsprechenden Datei enthält im Feld ID1ECB einen Fehlerschlüssel.

Auf Band kopierte ISAM-Dateien

ISAM-Dateien können zwar auf Band gesichert werden, sind als Banddateien jedoch nicht mit ISAM verarbeitbar.

Auf Bändern wird keine Information über das Dateiformat (BLKCTRL bzw. BLOCK-CONTROL-INFO) geführt. Diese Information erlischt also für auf Band (mit COPFILE bzw. mit COPY-FILE) kopierte Dateien, wenn der Katalogeintrag gelöscht wird.

Soll die Datei wieder zurückkopiert werden, muss dem COPFILE ein FILE-Aufruf mit dem Operanden STATE=FOREIGN vorausgehen bzw. dem Kommando COPY-FILE das Kommando IMPORT-FILE. Sie müssen den Operanden BLKCTRL bzw. BLOCK-CONTROL-INFO richtig versorgen, d.h. dem tatsächlichen Dateiformat entsprechend PAMKEY oder DATA bzw. *PAMKEY oder *WITHIN-DATA-BLOCK angeben.

Wird eine K-ISAM-Datei (BLKCTRL=PAMKEY) auf diese Weise – versehentlich – als NK-ISAM-Datei (BLKCTRL=DATA) kopiert, ist die entstehende Plattendatei nicht lesbar, da die ersten 16 Bytes einer jeden PAM-Seite, die bei BLKCTRL=PAMKEY Daten enthalten, mit Verwaltungsinformationen überschrieben werden.

OPEN-Fehler bei der Verarbeitung von NK-ISAM-Dateien

- NK-ISAM-Datei auf Band: Beim Importieren/Zurückschreiben der Banddatei wurde ein falsches Datenformat im FILE- oder FCB-Makroaufruf bzw. im Kommando IMPORT-FILE angegeben.
- ISAM-Pool überlastet: Ist der ISAM-Pool, in dem eine NK-ISAM-Datei eröffnet werden soll, überlastet, wird die OPEN-Verarbeitung mit dem Fehlercode `DMS0D9B` abgewiesen.

21.7 Umstellung von K-ISAM auf NK-ISAM und umgekehrt

- Migration
- Rückstieg
- Kompatibilität

21.7.1 Migration

Bei der Migration von K-ISAM nach NK-ISAM können drei Stufen unterschieden werden: Konvertierung, Änderung von Kommando-Prozeduren, Source-Code-Änderung.

Konvertierung

K-ISAM-Dateien können auf das NK-ISAM-Dateiformat durch „logisches Kopieren“ umgesetzt werden. „Logisches Kopieren“ ist z.B. möglich mit dem Dienstprogramm PERCON (siehe Handbuch „PERCON“ [12]). In diesen Fällen müssen weder die Programme verändert werden, die diese Dateien verarbeiten, noch die Kommandofolgen der Jobs, in denen diese Dateien angesprochen werden.

Kommando-Prozeduren

Über Änderungen der Kommandofolge in Prozeduren können die Performance-Vorteile genutzt werden, die die NK-ISAM-Schnittstelle mit den ISAM-Pools bietet. An geeigneter Stelle können in performance-kritischen Jobs die Kommandos zur ISAM-Pool-Verwaltung eingefügt werden. In FILE/FCB ist jeweils der Operand POOLLNK bzw. im Kommando ADD-FILE-LINK der Operand POOL-LINK zu ergänzen, damit die Zuordnung Datei/ISAM-Pool durchgeführt und die Datei im Pool verarbeitet werden kann. Sollen NK-ISAM-Dateien mit SHARUPD=YES bzw. SHARED-UPDATE=*YES eröffnet werden, müssen sie in hostspezifischen Benutzer-ISAM-Pools verarbeitet werden. Zu diesem Zweck muss vorher mit CREATE-ISAM-POOL der Auftrag mit dem entsprechenden ISAM-Pool verbunden werden. Mit ADD-ISAM-POOL-LINK muss dem Pool ein Kettungsname zugeordnet werden, der bei POOL-LINK anzugeben ist. Nach Abschluss der Dateiverarbeitung müssen die Bezüge auf den ISAM-Pool wieder abgebaut werden (REMOVE-ISAM-POOL, DELETE-ISAM-POOL), sofern nicht mit Standard-Pools gearbeitet wird.

Source-Code-Änderung

Über Änderungen im Source-Code der verarbeitenden Programme (und damit evtl. der Dateistruktur) kann die ISAM-Verarbeitung bzgl. NK-ISAM optimiert werden:

- Satzlänge an BLKSIZE anpassen, damit Überlaufblöcke vermieden werden
- in Flag-Anwendungen den zu untersuchenden Bereich möglichst klein halten (mit GETFL ...,LIMIT=)
- ISAM-Pool-Makroaufrufe sowie FILE-/FCB-Makroaufruf mit Operand POOLLNK für die ISAM-Verarbeitung im ISAM-Pools

21.7.2 Rückstieg

Die Umsetzung von NK-ISAM-Dateien in K-ISAM-Dateien kann über das Dienstprogramm PAMCONV erfolgen. Das Dienstprogramm PAMCONV ist im Handbuch „Dienstprogramme“ [13] beschrieben.

21.7.3 Kompatibilität

Da die ISAM-Aktionsmakros invariant sind gegenüber dem zu verarbeitenden Dateiformat, ergeben sich auf dieser Ebene keinerlei Umstellungsprobleme. Derselbe Makroaufruf kann in ein und demselben Programm z.B. sowohl auf NK-ISAM- als auch auf K-ISAM-Dateien angewendet werden, wenn die FCB-Adresse in einem Register übergeben wird. Die einzige Einschränkung an der Benutzerschittstelle ist die, dass Überlaufblöcke keine Teile des ISAM-Index enthalten dürfen. Beim Übergang von NK-ISAM nach K-ISAM muss natürlich beachtet werden, dass ISAM-Pool-Makros nur unterstützt werden, wenn NK-ISAM geladen ist.

Die folgende Tabelle beschreibt die wichtigsten Änderungen im NK-ISAM mit Hinweisen auf mögliche Auswirkungen.

NK-ISAM	Auswirkung
Flag-Verarbeitung	Performance-Einbußen beim GETFL-Makro
Index-/Daten-Trennung	wird ignoriert; die Sekundärzuweisung wird nur für den Datenteil wirksam
Dateiformat	bei UPAM-Verarbeitung von ISAM-Dateien zu berücksichtigen; nutzbarer Bereich in Datenblöcken kleiner; Überlaufblöcke erhöhen die I/O-Rate
ISAM-Pools	evtl. Adressraum-Mehrbedarf;Entlastung des Klasse-4-Speichers bei Shared-Update-Verarbeitung
Ein-/Ausgabebereiche	IOAREA1/2 durch Pools ersetzt; wird nur noch im Locate Mode benötigt und ist sinnvoll bei sequenzieller Verarbeitung
PAD-Faktor	PAD-Grenze wird überschritten, bevor ein neuer Datenblock angefordert wird
Satzsperre	ersetzt die Blocksperre
Checkpoint /Restart	kein Checkpoint-Schreiben mit hostspezifischen oder benutzerspezifischen ISAM-Pools; eine Checkpoint-Datei mit NK-ISAM-Dateien kann nur dann beim Restart verwendet werden, wenn NK-ISAM geladen ist

Tabelle 61: Inkompatibilitäten bei Umstellung auf NK-ISAM

21.8 „Crash-Resistenz" von ISAM-Dateien

Eine Datei ist „crash-resistent“, wenn sie sich nach einem Systemabbruch (System Crash) in einem konsistenten Zustand befindet und alle in der Vergangenheit erfolgreich ausgeführten Operationen erhalten geblieben sind.

Schreiboperationen sind nach einem Systemabsturz entweder vollständig ausgeführt oder gar nicht. In beiden Fällen ist die Datei jedoch aus ISAM-Sicht konsistent.

Bei einem Systemabbruch können geöffnete Dateien nicht mehr ordnungsgemäß geschlossen werden, sodass der LASTPG-Zeiger nicht korrekt ist. Ein folgender OPEN erkennt dies und gibt dem Benutzerprogramm in einer Fehlerroutine (OPENC-Ausgang des EXLST-Makros) die Möglichkeit, den Fehler zu beheben (VERIF-Makroaufruf). Ist keine solche Fehlerroutine vorgesehen, wird das Programm mit einer entsprechenden Meldung (`DMS0DD1, DMS0D1A`) abgebrochen. Der Zeiger auf das logische Dateiende (Last Page Pointer) kann mit dem Makroaufruf VERIF bzw. mit dem Kommando REPAIR-DISK-FILES aktualisiert werden (siehe Abschnitt „Dateien rekonstruieren“). Crash-Resistenz – wie oben beschrieben – ist garantiert bei WROUT=YES bzw. WRITE-IMMEDIATE=*YES und bei PUT-Verarbeitung mit Ein-/Ausgabebereich im Programm.

*Crash-Verhalten bei WROUT=NO bzw. WRITE-IMMEDIATE=*NO*

Bei WROUT=NO bzw. WRITE-IMMEDIATE=*NO können die Änderungen verloren gehen, die noch nicht auf die Platte zurückgeschrieben wurden. Die Dateikonsistenz kann mit dem Makro VERIF, REPAIR=ABS bzw. dem Kommando REPAIR-DISK-FILES wiederhergestellt werden. Die Daten können aus Benutzersicht jedoch noch inkonsistent sein. Die Datenkonsistenz muss dann vom Benutzer wiederhergestellt werden.

PUT (sequenzielles Schreiben) mit Ein-/Ausgabebereich im Programm

Wenn für PUT-Aufrufe ein Ein-/Ausgabebereich benutzt wird, wird erst dann eine Schreiboperation ausgelöst, wenn der Puffer die dem PAD-Wert entsprechende Füllung erreicht hat. Bei einem Systemabsturz sind alle Sätze verloren, die zwar im Puffer standen, aber noch nicht auf Platte geschrieben wurden. Aus Benutzersicht sind die Daten nicht konsistent/nicht vollständig; aus Sicht von ISAM ist die Datei jedoch konsistent.

22 SAM - Sequential Access Method

Mit der Zugriffsmethode SAM werden Dateien sequenziell verarbeitet. Eingeschlossen ist das Aufsuchen eines Satzes, Aktualisieren des Satzes sowie das Rückschreiben in die Datei.

Da SAM eine satzorientierte Zugriffsmethode ist, übernimmt es für den Benutzer das Blocken, Entblocken und Puffern der Sätze. Wenn im Benutzerprogramm zwei Ein-/Ausgabebereiche zur Verfügung stehen, kann Wechselpufferbetrieb genutzt werden; bei nur einem Ein-/Ausgabebereich findet keine überlappende Verarbeitung statt.

Die Zugriffsmethode SAM arbeitet überwiegend geräteunabhängig und gestattet die Verarbeitung von Dateien auf Platten und Bändern; Magnetbandkassetten werden weitgehend wie Magnetbänder behandelt.

Blockformate

Mit der Zugriffsmethode SAM können Dateien unterschiedlicher Blockformate verarbeitet werden (siehe auch Abschnitt „Blockformate für Plattendateien“ und Abschnitt „Blockformate für Banddateien“). Mit dem Operanden BLKCTRL in den Makros FILE und FCB bzw. dem Operanden BLOCK-CONTROL-INFO im Kommando ADD-FILE-LINK kann gewählt werden, ob eine K- oder eine NK-Datei verarbeitet werden soll:

- BLKCTRL=PAMKEY bzw. BLOCK-CONTROL-INFO=*PAMKEY vereinbart eine K-SAM-Datei

  K-SAM-Dateien (Key-SAM-Dateien) haben das herkömmliche Blockformat „PAMKEY“: Sie sind dadurch gekennzeichnet, dass für jede PAM-Seite DVS-Verwaltungsinformation in einem eigenen (außerhalb der Seite gelegenen) PAM-Schlüssel geführt wird.
- BLKCTRL=DATA/NO bzw. BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK/*NO legt eine NK-SAM-Datei fest

  NK-SAM-Dateien (Nonkey-SAM-Dateien) haben das Blockformat „DATA“ oder „NO“: Sie enthalten keine gesonderten PAM-Schlüssel. Beim Blockformat „DATA“ wird die DVS-Verwaltungsinformation innerhalb der PAM-Seite in einem Blockkontrollfeld hinterlegt.
  Das Blockformat „NO“ existiert bei SAM nur für Banddateien. Im Blockformat „NO“ werden blockspezifische Verwaltungsinformationen nicht unterstützt.

Hinsichtlich des Funktionsumfanges gibt es keine Unterschiede zwischen K- und NK-SAM-Dateien. Bei der Planung von Dateien muss jedoch berücksichtigt werden, dass bei NK-SAM-Dateien ein Teil des Logischen Blockes für das Blockkontrollfeld benötigt wird und daher für Benutzerdaten nicht zur Verfügung steht (Näheres siehe in den folgenden Abschnitt „Logischer Block einer K-SAM-Datei“ und Abschnitt „Logischer Block einer NK-SAM-Datei“).

SAM-Funktionen

Mit SAM lassen sich Dateien im Übertragungsbetrieb und im Ortungsbetrieb verarbeiten. Im Übertragungsbetrieb übernimmt das System die Übertragung der Datensätze zwischen Puffern und Benutzer-Ein-/Ausgabebereich, im Ortungsbetrieb werden die Sätze direkt im Ein-/Ausgabepuffer bearbeitet, der Benutzer ist für die korrekte Adressierung seiner Datensätze selbst verantwortlich. Im Ortungsbetrieb ist über die Wiedergewinnungsadresse, die das DVS bei Dateierstellung im FCB führt, ein direkter Zugriff möglich (siehe Abschnitt „Wiedergewinnungsadresse“).

Für die Dateibearbeitung mit SAM gibt es die folgenden Aktionsmakroaufrufe:

Makro	Kurzbeschreibung
GET	Sequenziell lesen; die Sätze werden nacheinander bereitgestellt, SAM „erwartet“, dass die Sätze in der Folge benötigt werden, in der sie geschrieben wurden.

PUT	Sequenziell schreiben: Im Übertragungsbetrieb führen die logischen Routinen der Zugriffsmethode das Blocken der Sätze durch, sodass die Ausgabe auf den Datenträger solange verzögert wird, bis der Ausgabepuffer gefüllt ist; die Puffer werden vom System automatisch bedient. Im Ortungsbetrieb muss der Benutzer selbst für das Blocken sorgen.
PUTX	Einen aktualisierten Satz zurückschreiben (nur im Ortungsbetrieb bei Plattendateien).
RELSE	Schließt einen Datenblock ab, d.h. für Eingabedateien: beim nächsten GET wird der nächste Datenblock eingelesen; für Ausgabedateien: beim nächsten PUT wird der Pufferinhalt als Datenblock geschrieben, der nächste Satz wird zum ersten Satz im neuen Datenblock. (Dies ist im Ortungsbetrieb nötig, wenn der folgende Satz nicht mehr in den aktuellen Puffer passt).
SETL	In der Datei auf einen bestimmten Block und bestimmten Satz darin positionieren.
FEOV	Für Banddateien: Bandwechsel auslösen.

Tabelle 62: Aktionsmakros für SAM-Dateibearbeitung

Werden Dateien als Ausgabedateien eröffnet (OPEN OUTPUT/EXTEND), interpretiert
SAM jeden PUT- oder SETL-Makroaufruf als „EOF-Anzeige“. Der letzte PUT oder SETL vor einem CLOSE zeigt dem System somit automatisch das Dateiende an. Sollen ab einem bestimmten Satz alle folgenden Sätze gelöscht werden, kann mit dem SETL-Makroaufruf auf den gewünschten Punkt in der Datei positioniert und anschließend die Datei mit CLOSE geschlossen werden.

Für den Direktzugriff wird dem Benutzer eine Wiedergewinnungsadresse bereitgestellt. Das Format dieser Wiedergewinnungsadresse ist ausführlich im Abschnitt „Wiedergewinnungsadresse“ beschrieben. Beim Schreiben eines Satzes wird seine Wiedergewinnungsadresse im FCB bereitgestellt.
Der Benutzer kann, falls er es wünscht, aus den Daten der Wiedergewinnungsadresse eine neue Datei aufbauen und hierbei eine Grundlage für anschließende nicht-sequenzielle Verarbeitung der Datei schaffen, die gerade erstellt wird. Nach der Ausführung eines GET-Makroaufrufs wird diese Wiedergewinnungsadresse ebenfalls im FCB bereitgestellt. Auch hier kann der Benutzer- falls er seine Datei nicht selbst erstellt hat- eine zweite Datei aus den Wiedergewinnungsadressen aufbauen, um eine anschließende nichtsequenzielle Verarbeitung der Datei durchzuführen.

Wird im Übertragungsbetrieb mit zwei Ausgabepuffern beim Schreiben eines Datenblockes für eine Banddatei das physikalische Bandende erkannt, so wird der andere Puffer (der nur einen Satz enthält) noch auf das alte Band geschrieben. Erst dann wird der Bandwechsel eingeleitet.

Soll eine Datei im Ortungsbetrieb erstellt werden (OPEN OUTPUT/EXTEND), erhalten Sie nach Abschluss des OPEN in dem Register, das Sie im FCB-Operanden IOREG angegeben haben, die Anfangsadresse des ersten zu schreibenden Satzes. Nach Ausführung des PUT-Makroaufrufs enthält das IOREG-Register die Anfangsadresse des nächsthöheren Satzes.
Bei einer Datei mit Sätzen variabler Länge (RECFORM=V) erhalten Sie außerdem immer die Zahl der noch freien Byte im aktuellen Block: Sie wird in dem Register übergeben, das der Aufrufer im Operanden VARBLD des FCB-Makros angegeben hat.

22.1 Logische Blöcke von SAM-Dateien

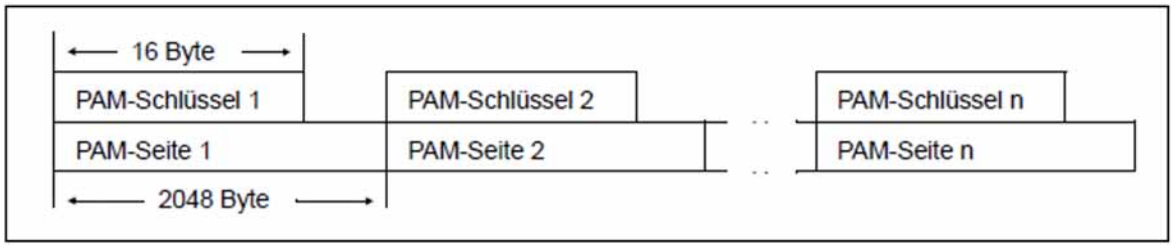
Das Format der logischen Blöcke wird in den Makros FILE und FCB mit dem Operanden BLKCTRL bzw. mit dem Operanden BLOCK-CONTROL-INFO im Kommando ADD-FILE-LINK festgelegt. SAM-Dateien können Sätze des Formats F, V oder U enthalten.

Die Satzlänge darf die Blocklänge nicht überschreiten. Die maximale Blocklänge bei Nichtstandardblöcken ist für SAM-Banddateien 32768 Byte.

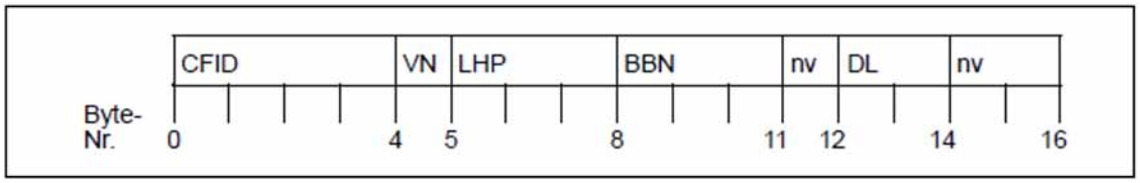
22.1.1 Logischer Block einer K-SAM-Datei

Jeder PAM-Seite einer herkömmlichen K-SAM-Datei ist ein außerhalb der Seite gelegener, 16 Byte langer PAM-Schlüssel zugeordnet, in dem DVS-Verwaltungsinformation geführt wird.

In einer K-SAM-Datei, bei der mit dem Operanden BLKSIZE=(STD,n) im FILE- oder FCB-Makro bzw. mit dem Operanden BUFFER-LENGTH=*STD(SIZE=n) im ADD-FILE-LINK-Kommando n PAM-Seiten zu einem Logischen Block zusammengefügt wurden, hat ein solcher Logischer Block demnach folgenden Aufbau:



Die n * 2048 Byte des Logischen Blockes stehen gänzlich für Benutzerdaten zur Verfügung, weil die DVS-Verwaltungsinformation in die 16 Byte langen PAM-Schlüssel ausgelagert ist. Bei SAM-Dateien hat der PAM-Schlüssel folgenden Aufbau:



Dabei bedeutet:

CFID	Coded File Identification: Kennzeichen, an dem das DVS durch Vergleich mit einer entsprechenden CFID im Katalogeintrag feststellen kann, ob die Daten der PAM-Seite zur Datei gehören oder ob diese Seite (für diese Datei) zwar reserviert ist, aber noch keine Daten enthält.
VN	Versionsnummer: Sie ermöglicht die partielle Sicherung einer Datei durch ARCHIVE oder HSMS. Dabei werden nur die PAM-Seiten einer Datei gesichert, die seit der letzten Voll- oder Differenzsicherung geändert wurden.
LHP	Logische Half-Page-Nummer: Sie wird zur Konsistenzprüfung einer Datei verwendet sowie zu ihrer Entzerrung, wenn sie von ARCHIVE oder HSMS aus einer Sicherung wiederhergestellt wird.
BBN	Binäre Blocknummer
DL	Länge der verwendbaren Daten im Puffer
nv	Von SAM nicht verwendete Byte des PAM-Keys

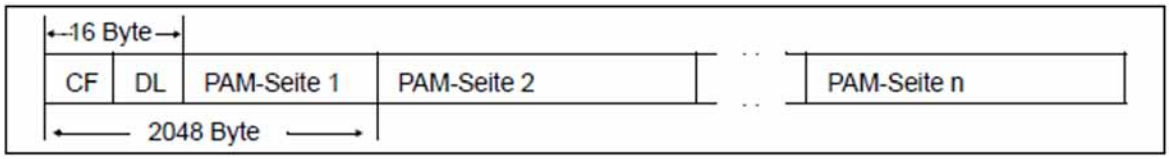
22.1.2 Logischer Block einer NK-SAM-Datei

Im Unterschied zu den herkömmlichen K-SAM-Dateien besitzen NK-SAM-Dateien außerhalb der PAM-Seiten keine gesonderten PAM-Schlüssel. Je nachdem, welches Blockformat für eine NK-SAM-Datei vereinbart ist, werden block-spezifische Verwaltungsinformationen entweder in den Logischen Blöcken selbst abgelegt oder überhaupt nicht geführt:

Blockformat „DATA“

Dieses Blockformat wird bei Erstellung der Datei mit dem Operanden BLKCTRL=DATA im Makro FILE bzw. mit dem Operanden BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK im Kommando ADD-FILE-LINK erzeugt. Die blockspezifische Verwaltungsinformation wird in einem Teil jedes logischen Blocks der Datei – dem 12 Byte langen so genannten Blockkontrollfeld (Control Field) – untergebracht. NK-SAM fügt diesem Blockkontrollfeld noch eine 4 Byte lange Information über die Länge der verwendbaren Daten im Puffer an.

In einer NK-SAM-Datei, bei der mit dem Operanden BLKSIZE=(STD,n) im FILE- oder FCB-Makro bzw. mit dem Operanden BUFFER-LENGTH=*STD(SIZE=n) im ADD-FILE-LINK-Kommando n (n <= 16) 2K-Blöcke (Standardblöcke) zu einem Logischen Block zusammengefügt wurden, ergibt sich demnach folgender Aufbau für einen solchen Logischen Block:

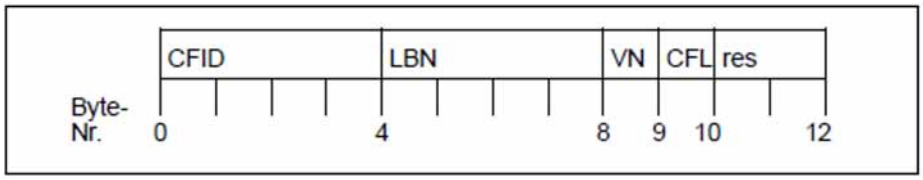


Dabei bedeutet:

CF Blockkontrollfeld (Control Field), 12 Byte lang

DL Länge der verwendbaren Daten im Puffer, 4 Byte lang

Das Blockkontrollfeld hat folgenden Aufbau:



Dabei bedeutet:

CFID Coded File IDentification: Gleiche Bedeutung wie im PAM-Schlüssel (siehe Abschnitt „Logischer Block einer K-SAM-Datei“).

LBN Logical Block Number (Logische Blocknummer): Nummer des Standardblockes, der das Blockkontrollfeld enthält. (Standardblöcke werden innerhalb der Datei fortlaufend nummeriert).

VN Versionsnummer: Gleiche Bedeutung wie im PAM-Schlüssel.

CFL Control FLag: Das Control Flag kennzeichnet Lücken in einer Datei. Diese Information wird vor allem beim Kopieren einer Datei vom bzw. auf Band ausgewertet.

res Für künftige Erweiterungen reservierte Byte des Blockkontrollfeldes.

Der vom 12 Byte langen Blockkontrollfeld und der sich anschließenden 4 Byte langen SAM-spezifischen Füllgradinformation beanspruchte Teil eines Logischen Blockes steht dem Benutzer nicht mehr für seine Daten zur Verfügung. Diesem Umstand muss er bei der Planung der Blockungsfaktoren und Satzlängen für ein NK-SAM-Datei Rechnung tragen.

Für die Länge des für Benutzerdaten maximal nutzbaren Teils eines Logischen Blockes aus n Standardblöcken (maximal zulässige Satzlänge) gilt demnach – unabhängig vom vereinbarten Satzformat – folgende Formel:

Nutzbare Blocklänge

```
    RECSIZE <= n * 2048 - 16
```

`RECSIZE`	zulässige Satzlänge in der NK-SAM-Datei
n	Blockungsfaktor aus BLKSIZE = (STD,n), 1 <= n <= 16

Dateien des Blockformats „DATA“ sind bei Anwendung der Pufferverschiebung (Operand BUFOFF=16 im FCB und FILE-Makro bzw. Operand BLOCK-OFFSET=16 im Kommando ADD-FILE-LINK) datenaustauschfähig. Sie können in die Version 9.5 importiert werden.

Blockformat „NO“

Dieses Blockformat wird bei Erstellung der Datei mit dem Operanden BLKCTRL=NO im Makro FILE bzw. mit dem Operanden BLOCK-CONTROL-INFO=*NO im Kommando ADD-FILE-LINK erzeugt. Dieses Blockformat wird bei SAM *nur für Banddateien* unterstützt. Es werden keine block-spezifischen Verwaltungsinformationen angelegt. Dem Benutzer steht der gesamte Logische Block für Nutzdaten zur Verfügung.

Eine Banddatei, die von anderen Systemen mit block-spezifischen Verwaltungsinformationen erstellt worden ist, kann mit einer geeigneten Pufferverschiebung (Operanden BUFOFF / BLKCTRL=NO in den Makros FCB und FILE bzw. den Operanden BLOCK-OFF-SET / BLOCK-CONTROL-INFO=*NO im Kommando ADD-FILE-LINK) importiert werden.

22.2 Umstellung von K-SAM auf NK-SAM und umgekehrt

Umstellung von K-SAM auf NK-SAM

Voraussetzung

K-SAM-Dateien können mithilfe der Dienstprogramme PAMCONV (siehe Handbuch „Dienstprogramme“ [13]) oder PERCON (siehe Handbuch „PERCON“ [12]) in NK-SAM-Dateien umgesetzt werden. Voraussetzung dafür ist, dass die maximale Satzlänge in der K-SAM-Datei nicht größer ist als die um 16 Byte reduzierte nutzbare Blocklänge der NK-SAM-Datei. Bei einer K-SAM-Datei mit BLKSIZE=(STD,n) bzw. BUFFER-LENGTH= STD(SIZE=n) muss daher folgende Bedingung erfüllt sein:

maximale Satzlänge

$$RECSIZE_{K\text{-}SAM} <= (n * 2048 - 16)$$

$RECSIZE_{K\text{-}SAM}$	maximale Satzlänge in der K-SAM-Datei
n	Blockungsfaktor aus BLKSIZE = (STD,n), 1 <= n <= 16

Optimierung der Blocknutzung

K-SAM-Dateien, die zwar die oben beschriebene Voraussetzung hinsichlich der Satzlänge erfüllen, jedoch alle Byte des Logischen Blocks im K-SAM-Format für Benutzerdaten nutzen, benötigen nach ihrer Umsetzung in das NK-SAM-Format möglicherweise erheblich (bis zu 100%) mehr Speicherplatz, da wegen der reduzierten nutzbaren Blocklänge im NK-SAM-Format weniger Datensätze in einen Logischen Block passen. Diese reduzierte nutzbare Blocklänge ist zu berücksichtigen, wenn im NK-SAM-Format alle verfügbaren Byte eines Logischen Blocks für Benutzerdaten genutzt werden sollen (siehe "Logischer Block einer NK-SAM-Datei"):

nutzbare Blocklänge

$$NBL_{N\ K\text{-}SAM} <= (n * 2048 - 16)$$

$NBL_{NK\text{-}SAM}$	(für Benutzerdaten) nutzbare Blocklänge in der NK-SAM-Datei
n	Blockungsfaktor aus BLKSIZE = (STD,n), 1 <= n <= 16

Speicherplatzbedarf

Bei der Umstellung vom K-SAM- auf das NK-SAM-Format vergrößert sich der Platzbedarf einer Datei mit Sätzen variabler Länge um durchschnittlich 1%.

Der Speicherplatzzuwachs einer Datei mit Sätzen fester Länge bei der Umsetzung vom K-SAM- auf NK-SAM-Format lässt sich anhand folgender Formel berechnen:

Speicherplatzbedarf einer NK-SAM-Datei

$$\#PAM\text{–}Seiten_{NKSAM} = (n * 2048 / (n * 2048 - 16)) * \#PAM\text{–}Seiten_{K\text{-}SAM}$$

$\#PAM\text{-}Seiten_{NK\text{-}SAM}$	Speicherbedarf der NK-SAM-Datei in PAM-Seiten
$\#PAM\text{-}Seiten_{K\text{-}SAM}$	Speicherbedarf der K-SAM-Datei in PAM-Seiten
n	Blockungsfaktor aus BLKSIZE = (STD,n), 1 <= n <= 16

Umstellung von NK-SAM auf K-SAM

NK-SAM-Dateien können mit dem Dienstprogramm PAMCONV (siehe Handbuch „Dienstprogramme“ [13] oder dem Software-Produkt PERCON (siehe Handbuch „PERCON“ [12]) in K-SAM-Dateien umgesetzt werden.

Kompatibilität

NK-SAM bietet dem Benutzer den gleichen Funktionsumfang wie K-SAM mit identischen Schnittstellen.

Beim Übergang von K-SAM zu NK-SAM können sich allerdings Änderungen an Programmen als nötig erweisen, die Wiedergewinnungsadressen für Datensätze in einem eigenen Algorithmus berechnen, da sich bei der Umsetzung die Blockung der zu verarbeitenden Datei ändern kann: Wegen der geringeren nutzbaren Blocklänge passt möglicherweise ein Satz weniger in jeden Block, und die Datei wird größer. In diesem Fall muss auch der Berechnungsalgorithmus für die Wiedergewinnungsadresse geändert werden.

In Anwendungen, die die vom DVS im FCB zur Verfügung gestellte Wiedergewinnungsadresse auswerten (siehe Abschnitt „Wiedergewinnungsadresse“), sind dagegen keine Änderungen erforderlich.

22.3 SAM-Datei eröffnen (OPEN-Modi)

In der folgenden Tabelle sind die OPEN-Modi zusammengestellt.

Open-Modus	Kurzbeschreibung
INPUT	Sequenzielles Lesen in Richtung Dateiende; die Datei muss existieren
OUTPUT	Neue Datei sequenziell erstellen oder eine vorhandene Datei überschreiben
EXTEND	Datei erweitern
UPDATE	Nur für Plattendateien im Ortungsbetrieb: Sätze aktualisieren; der zu aktualisierendeSatz muss mit GET bereitgestellt werden, bei der Verarbeitung darf die Satzlänge nicht verändert werden, der aktualisierte Satz wird mit PUTX zurückgeschrieben.
REVERSE	Sequenzielles Lesen in Richtung Dateianfang; die Datei muss existieren; Banddateien, die sich über mehrere Datenträger erstrecken, können nur dateiabschnittsweise mithilfe des Operanden VSEQ (Makro FILE) bzw. des Operanden START-POSITION (Kommando ADD-FILE-LINK) gelesen werden; kein automatischer Bandwechsel.

Tabelle 63: Open-Modi bei Eröffnen einer SAM-Datei

Die folgende Tabelle gibt Auskunft darüber, welche Aktionsmakros in Verbindung mit den o.g. Open-Modi zulässig sind.

Aktions-Makro	OPEN-Typ				
	INPUT	OUTPUT	EXTEND	UPDATE	REVERSE
GET	x			x	x
PUT		x	x		
PUTX				x	
RELSE	x	x	x	x	x
SETL	x	x	x	x	x

Tabelle 64: SAM-Aktionsmakroaufrufe und OPEN-Modi

Bei der Bearbeitung von SAM-NODE-FILES muss eine Anwendung beim Öffnen der Datei im Dateisteuerblock (FCB) den Indikator SAM_NODE_FILE_ENABLE setzen um anzuzeigen, dass sie die Besonderheiten bei der Verarbeitung von SAM-Node-Files kennt (siehe Hinweis im Abschnitt „Wiedergewinnungsadresse“).

Primär-/Sekundärzuweisung bei Plattendateien

Wird eine SAM-Datei erstellt oder erweitert (OPEN OUTPUT/EXTEND), müssen Primär- und Sekundärzuweisung mindestens gleich der Blockgröße sein.

Soll im Übertragungsbetrieb eine Datei erstellt oder erweitert werden (OPEN OUTPUT/EXTEND), die mit RECFORM=F oder RECFORM=V definiert wurde und pro Datenblock mehr als einen Satz enthält, so gilt:

- Die Primärzuweisung muss mindestens das Doppelte der Datenblocklänge betragen (Primärzuweisung >= 2 * BLKSIZE). Andernfalls geht die Steuerung an den EXLST-Ausgang NOSPACE (Speicherplatzzuweisung nicht ausreichend).
- Die Sekundärzuweisung durch SAM wird bereits beim ersten Satz des letzten Blockes eingeleitet, der noch in den zugewiesenen Bereich geschrieben werden kann. Kann die Sekundärzuweisung nicht durchgeführt werden, dann geht die Steuerung an den EXLST-Ausgang NOSPACE (sofern er im Programm vorgesehen ist). Dies gibt dem Benutzer die Möglichkeit, die restlichen Sätze des letzten Blockes noch zu schreiben.

Auswirkungen des LABEL-Operanden bei Banddateien

Die folgende Tabelle zeigt die Auswirkungen des Makro- bzw. Kommando-Operanden beim Öffnen einer SAM-Datei:

Makro FILE: Operand LABEL=	**Kommando ADD-FILE-LINK: Operand LABEL-TYPE=**	**Auswirkung**
NO / NSTD	*NO / *NON-STD	Angaben im BUFOFF-Operanden führen zum OPEN-Fehler bei CODE=ISO/OWN und BLK-SIZE=STD
(STD,0)	*STD(DIN-REVISION-NUMBER=0)	Angaben im BUFOFF-Operanden und CODE=ISO/OWN führen zum OPEN-Fehler
(STD,1)	*STD(DIN-REVISION-NUMBER=1)	Angaben im BUFOFF-Operanden und die Angabe von CODE=ISO/OWN führen zum OPEN-Fehler;
(STD,2)	*STD(DIN-REVISION-NUMBER=2)	Standardblöcke werden in Nichtstandardblöcke(BLKCTRL=DATA/NO) und V-Sätze in D-Sätze umgewandelt; RECSIZE > 9999 führt zusammen mit RECFORM=V zum OPEN-Fehler
(STD,3)	*STD(DIN-REVISION-NUMBER=3)	Standardblöcke werden in Nichtstandardblöcke(BLKCTRL=DATA/NO) und V-Sätze in D-Sätze umgewandelt; RECSIZE > 9999 führt zusammen mit RECFORM=V zum OPEN-Fehler
STD oder keine Angabe	*STD	Angaben im BUFOFF-Operanden führen zum OPEN-Fehler.

Aus der Kombination der Angaben in den Operanden CODE, RECSIZE und RECFORM (Makro FILE) bzw. CODE, RECORD-SIZE und RECORD-FORMAT (Kommando ADD-FILE-LINK) ergeben sich folgende, implizite Werte für LABEL, wenn die zu erstellende Datei die Erste auf dem Band ist:

CODE	**Blockformat**	**REFORM**	**LABEL-Wert (implizit)**
EBCDIC	PAMKEY	-	(STD,1)

EBCDIC	DATA/NO	U	(STD,2)
EBCDIC	DATA/NO	F/V	(STD,3), V-Sätze --> D-Sätze
ISO/OWN	PAMKEY	-	OPEN-Fehler
ISO/OWN	DATA/NO	U	(STD,2)
ISO/OWN	DATA/NO	F	(STD,3)
ISO/OWN	DATA/NO	V	V-Sätze --> D-Sätze (STD,3) OPEN-Fehler falls BLKSIZE > 9999

Tabelle 66: Auswirkungen von CODE-, BLKSIZE- und RECFORM-Angaben auf den LABEL-Operanden

Die Kombination von BUFOFF und RECFORM=U oder BLKCTRL=DATA im Makro FILE ist nicht zulässig. Ebenso ist die Kombination der Operanden RECORD-FORMAT=*UNDEFINED und BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK nicht zulässig!

Bei LABEL=(STD,1) bzw. LABEL-TYPE=*STD(DIN-REVISION=1) ist es nicht möglich, eine Datei mit CODE=EBCDIC, mit Nicht-Standardblöcken und D-Sätzen zu schreiben, sie kann jedoch gelesen werden.

Es kommt zum OPEN-Fehler, wenn ein im LABEL-Operanden geforderte Standardkennsatzstufe nicht mit der im VOL1-Kennsatz enthaltenen übereinstimmt.

Wurde LABEL=STD bzw. LABEL-TYPE=*STD angegeben oder keine Angabe gemacht, wird die Kennsatzstufe entsprechend der o.g. Tabelle 66 ermittelt. Liegt die so ermittelte Kennsatzstufe über der im VOL1-Kennsatz, gilt die im VOL1-Kennsatz definierte Kennsatzstufe. Liegt sie unter der Kennsatzstufe im VOL1-Kennsatz oder gilt aus dem VOL1-Kennsatz LABEL=(STD,0) und gilt CODE=ISO/OWN, kommt es zum OPEN-Fehler.

22.4 Satzformate für SAM-Dateien

Bei SAM sind die Satzformate F (feste Satzlänge), V (variable Satzlänge) und U (undefinierte Satzlänge) zulässig.

Bei Format U schreibt/liest SAM pro Datenblock (Puffer) nur einen Satz. Die Definition RECFORM=U in Verbindung mit BLKSIZE=STD und BLKCTRL=PAMKEY (entspricht RECORD-FORMAT=*UNDEFINED in Verbindung mit BUFFER-LENGTH=*STD und BLOCK-CONTROL-INFO im Kommando ADD-FILE-LINK) sowie einer aktuellen Satzlänge von 48 Byte hätte z.B. zur Folge, dass in jeder PAM-Seite 2000 Byte „verschwendet“ würden.

Wird ein Standardblock nicht voll ausgenutzt (z.B. nach den Aktionsmakroaufrufen RELSE, SETL, FEOV oder CLOSE), bleiben die restlichen Byte unverändert; d.h. sie haben undefinierten Inhalt.

Die Satzlänge darf die Blocklänge nicht überschreiten.

Weitere Angaben zum Satzformat sind dem Abschnitt „Satzformate“ zu entnehmen.

22.5 Wiedergewinnungsadresse

Das DVS versorgt beim Erstellen einer SAM-Datei im FCB eine Wiedergewinnungsadresse, die für direkten Zugriff bei späterer Verarbeitung genutzt werden kann. Die Wiedergewinnungsadresse besteht aus Block- und Satznummern, die Blocknummer bezieht sich immer auf den logischen Datenblock (nicht auf PAM-Seiten), die Satznummer zeigt die Position des Satzes innerhalb eines Datenblocks. Bei Mehrbanddateien ist zu beachten, dass die Blocknummer nur innerhalb eines Bandes geführt wird.

Im 31-bit-TU-FCB, d.h. bei XS-Programmierung, ist die Wiedergewinnungsadresse auf zwei jeweils ein Wort lange Felder aufgeteilt: das Feld ID1BLK# enthält die Blocknummer innerhalb der Datei, das Feld ID1REC# die Satznummer innerhalb des Datenblocks. Satz- und Blockzähler werden bei PUT und GET vom System automatisch hochgezählt. Wird eine Datenübertragung ausgelöst, wird der Satzzähler automatisch zurückgesetzt.

Bei Nicht-XS-Verarbeitung ist die Wiedergewinnungsadresse im 24-bit-TU-FCB im Feld ID1RPTR in der Form „bbbbbbrr“ enthalten:

- „bbbbbb“ ist die Nummer des Datenblocks in der Datei
- „rr“ die Nummer des Satzes innerhalb des Datenblocks

Der Satzzähler wird vom System nicht automatisch hochgezählt, dies muss der Benutzer im Programm durchführen, wenn er die Wiedergewinnungsadresse nutzen will. Der Satzzähler wird jedoch bei jeder Datenübertragung automatisch zurückgesetzt.
Für Bandverarbeitung ist zu beachten, dass die Wiedergewinnungsadresse bei Nicht-XS-Verarbeitung nur für Dateien mit Standardblockung versorgt wird, sodass für Dateien mit Nicht-Standardblöcken kein SETL R möglich ist.

XS-Schnittstelle: 31-bit-TU-FCB		**Nicht-XS-Schnittstelle: 24-bit-TU-FCB**	
ID1BLK# (1 Wort)	ID1REC# (1 Wort)	ID1RPTR (1 Wort)	
		Byte 1-3	Byte 4
Blocknumer	Satznummer	Blocknummer	Satznummer

Tabelle 67: Aufbau der Wiedergewinnungsadresse

Der erste Satz einer Datei hat also folgende Wiedergewinnungsadressen:

- im 31-bit-TU-FCB 00000001 im Feld ID1BLK# und 00000001 im Feld ID1REC#
- im 24-bit-TU-FCB 00000101 im Feld ID1RPTR

Die folgende Tabelle zeigt die Werte der Wiedergewinnungsadresse nach den Aktionsmakros SETL B und SETL E in Abhängigkeit des Open-Modus

OPEN-Modus	**SETL B**			**SETL E**		
	ID1BLK#	ID1REC#	ID1RPTR	ID1BLK#	ID1REC#	ID1RPTR
INPUT, UPDATE	-	-	-	max	1	max 1
OUTPUT	1	0	1 0	Fehler	Fehler	Fehler
EXTEND	1	0	1 0	Fehler	Fehler	Fehler
REVERSE	-	-	-	-	-	-

Tabelle 68: Werte der Wiedergewinnungsadresse nach SETL B und SETL E, abhängig von OPEN

- Feldinhalt unverändert

max höchste Blocknummer

Im Feld IDRPTR sind Block- und Satzzähler dargestellt.

Die folgende Tabelle erläutert, in welcher Weise die Aktionsmakroaufrufe die Wiedergewinnungsadresse versorgen:

Makro	Versorgung der Wiedergewinnungsadresse
GET	Wird durch den angegebenen Satz eine Datenübertragung erforderlich, enthält der Blockzähler die logische Blocknummer, der Satzzähler wird zurückgesetzt. Bei XS-Programmierungwird der Satzzähler bei jedem Aktionsmakroaufruf aktualisiert.
PUT	Wird durch den PUT-Makroaufruf für den angegebenen Satz eine Datenübertragung ausgelöst, dann wird der Blockzähler aktualisiert und der Satzzähler zurückgesetzt. Das heißt, dass der Blockzähler auf die Nummer des neuen Datenblocks gesetzt wird, der den Satz aufnehmen soll. Bei XS-Programmierung wird der Satzzähler bei jedem Aktionsmakroaufrufaktualisiert.
RELSE	Wenn eine Datei erstellt oder erweitert wird (OPEN OUTPUT/EXTEND) enthält der Blockzähler die Nummer des Datenblocks, der den folgenden Satz aufnehmen soll, der Satzzähler wird auf null gesetzt.
FEOV	nach Bandwechsel werden Block- und Satzzähler für das neue Band vom System zurückgesetzt.

Tabelle 69: SAM-Aktionsmakros und Wiedergewinnungsadresse

Beispiel

Die Dateieigenschaften werden definiert :

- über den Makro FILE:
  BLKCTRL=PAMKEY,
  BLKSIZE=(STD,2),
  RECFORM=F,
  RECSIZE=512

oder

- mit dem Kommando ADD-FILE-LINK:
  BLOCK-CONTROL-INFO=*PAMKEY,
  BUFFER-LENGTH=*STD(SIZE=2),
  RECORD-FORMAT=*FIXED,
  RECORD-SIZE=512

Die Wiedergewinnungsadresse lautet:

	Wiedergewinnungsadresse		
	31-bit-TU-FCB		24-bit-TU-FCB
	ID1BLK#	ID1REC#	IDRPTR
für Satz 10	00000002	00000002	00000202
für Satz 20	00000003	00000004	00000304

Hinweis zur Bearbeitung von SAM-Node-Files

Wird beim Beschreiben eines SAM-Node-Files ein Datenblock mit RELSE abgeschlossen, bevor er vollständig gefüllt worden ist, sind die Wiedergewinnungsadressen ab dieser Stelle nur gültig solange die Datei noch geöffnet ist.
Nach CLOSE und erneutem OPEN, ergibt sich eine andere Aufteilung der Datensätze des Node-Files auf die SAM-Blöcke, die der Zugriffsmethode zur Bearbeitung übergeben werden, so dass die vorherigen Wiedergewinnungsadressen nicht mehr zur Datei passen!

Der gleiche Effekt tritt bei der OPEN UPDATE-Verarbeitung auf. Wurde ein SAM-Node-File im UPDATE-Modus geöffnet und erweitert, sind Wiedergewinnungsadressen nach Schließen und erneutem Öffnen der Datei unbrauchbar.

Dieses Verhalten bei SAM-Node-Files (Net-Storage) ist inkompatibel zur Verarbeitung von SAM-Dateien auf herkömmlichem Public-Space, wo die Block- und Satzstruktur auch nach CLOSE und OPEN erhalten bleibt. Die Anwendung muss daher vor dem Öffnen der Datei mit dem Indikator SAM_NODE_FILE_ENABLE im Dateisteuerblock (FCB) anzeigen, dass sie in der Lage ist SAM-Node-Files korrekt zu verarbeiten.

23 UPAM - User Primary Access Method

UPAM ist die primäre blockorientierte Zugriffsmethode im BS2000 für wahlfreien Zugriff auf Platten- und Banddateien. Zu jedem Zeitpunkt kann auf einen beliebigen Block der Datei lesend oder schreibend zugegriffen werden.

Mit der Zugriffsmethode UPAM können Dateien unterschiedlicher Blockformate verarbeitet werden (siehe Abschnitt „Blockformate für Plattendateien“ und Abschnitt„Blockformate für Banddateien“):

- K-PAM-Dateien (Key-PAM-Dateien) haben das Blockformat „PAMKEY“: Sie sind dadurch gekennzeichnet, dass für jede PAM-Seite DVS-Verwaltungsinformation in einem eigenen (außerhalb der Seite gelegenen) PAM-Schlüssel geführt wird.
- NK-PAM-Dateien (Nonkey-PAM-Dateien) haben das Blockformat „DATA“ oder „NO“. Beim Blockformat „DATA“ wird die DVS-Verwaltungsinformation innerhalb des Datenblockes in einem Blockkontrollfeld hinterlegt. Beim Blockformat „NO“ werden keinerlei block-spezifische Verwaltungsinformationen hinterlegt.
  Es gibt NK2-PAM-Dateien und NK4-PAM-Dateien. Näheres hierzu siehe Abschnitt„Blockformate für Plattendateien“.

Mit dem Operanden BLKCTRL in den Makros FILE und FCB können Sie wählen, ob eine K- oder eine NK-Datei verarbeitet werden soll: BLKCTRL=PAMKEY vereinbart eine K-PAM-Datei, BLKCTRL=DATA oder BLKCTRL=NO legt eine NK-PAM-Datei fest.

Auf Kommandoebene kann mit dem Operanden BLOCK-CONTROL-INFO im Kommando ADD-FILE-LINK festgelegt weden, ob eine K- oder eine NK-Datei verarbeitet werden soll: BLOCK-CONTROL-INFO=*PAMKEY vereinbart eine K-PAM-Datei, BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK oder BLOCK-CONTROL-INFO=*NO legt eine NK-PAM-Datei fest.

Auf einer NK2-Platte wird bei Angabe der Blockgröße mit BLKSIZE=(STD,n) bzw. BUFFER-LENGTH=*STD (SIZE=n), wobei n eine gerade Zahl ist, eine NK4-PAM-Datei angelegt; ist n eine ungerade Zahl wird eine NK2-PAM-Datei angelegt.
Auf einer NK4-Platte kann nur eine NK4-PAM-Datei liegen (siehe auch Abschnitt „Plattenformate“).

Grundlage der Verarbeitung ist der Datenblock; Satzstrukturen werden nicht erkannt. Sowohl bei K-PAM-Dateien (Blockformat BLKCTRL=PAMKEY) als auch bei NK-PAM-Dateien (Blockformat BLKCTRL=DATA oder BLKCTRL=NO) besteht der Datenblock (logischer Block) aus einem oder mehreren 2048-Byte-Standardblöcken (Angabe BLKSIZE=(STD,n)).

Durch die Angabe BLKSIZE=(STD,n) (n > 1) im FCB- oder FILE-Makro bzw. durch BUFFER-LENGTH=*STD (SIZE=n) im Kommando ADD-FILE-LINK können mehrere 2048-Byte-Standardblöcke zu einem Datenblock (logischen Block) zusammengefasst werden.

PAM-Dateien vom Typ Node-File haben stets das Blockformat BLKCTRL=NO.

Bei einer K-PAM-Datei kann jeder Standardblock innerhalb des logischen Datenblockes im Programm angesprochen werden.
Bei einer NK-PAM-Datei kann nur der gesamte Datenblock (logischer Block) im Programm angesprochen werden; eine getrennte Verarbeitung der einzelnen 2048-Byte-Blöcke, aus denen er gebildet wurde, ist nicht möglich.

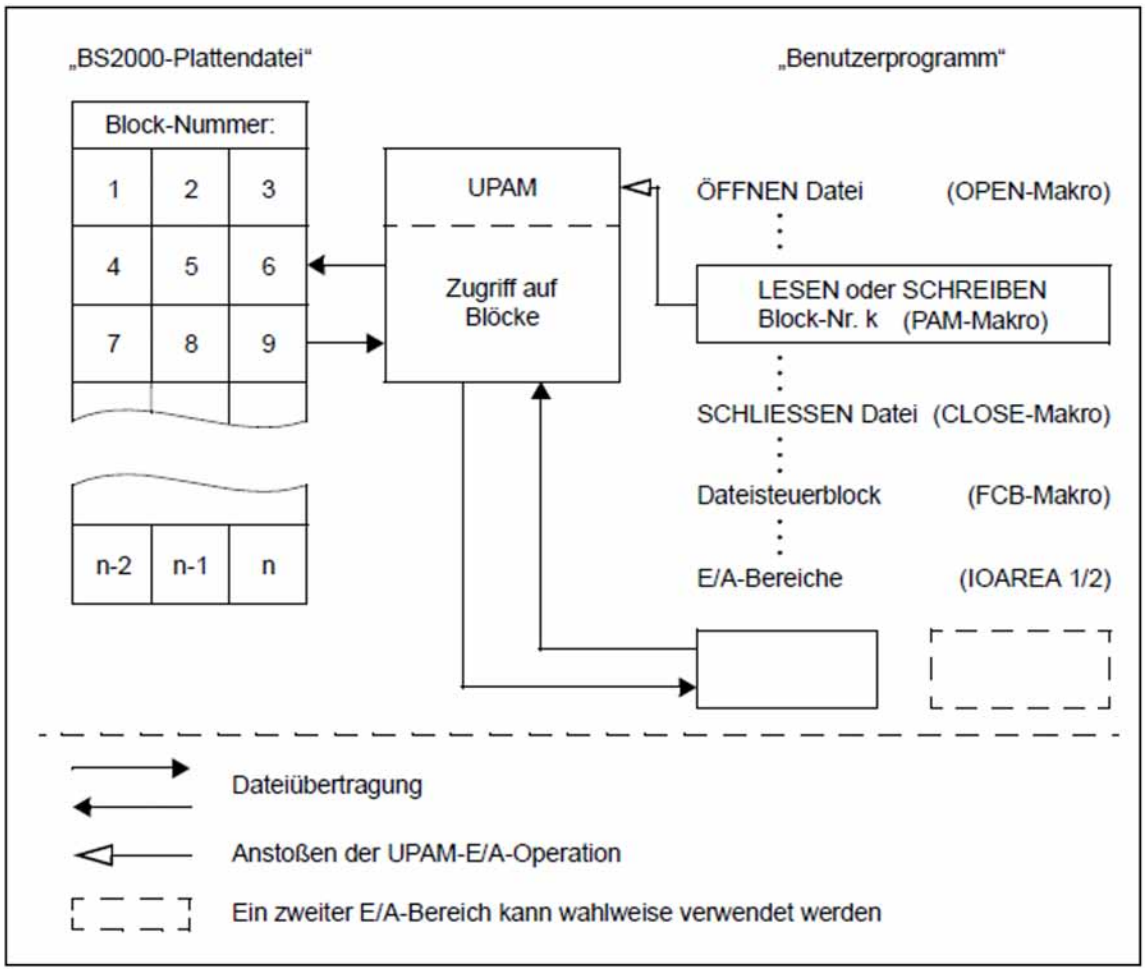


Bild 39: Arbeitsweise von UPAM

23.1 Datenblöcke

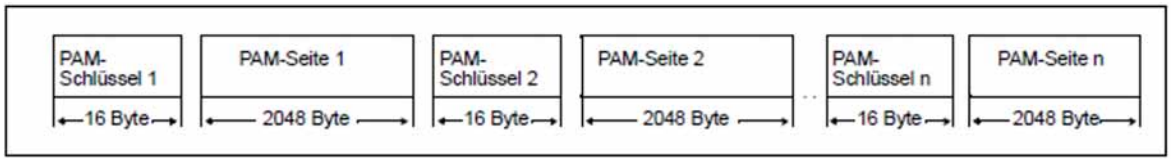
- Datenblock einer K-PAM-Datei
- Logischer Block einer NK-PAM-Datei

23.1.1 Datenblock einer K-PAM-Datei

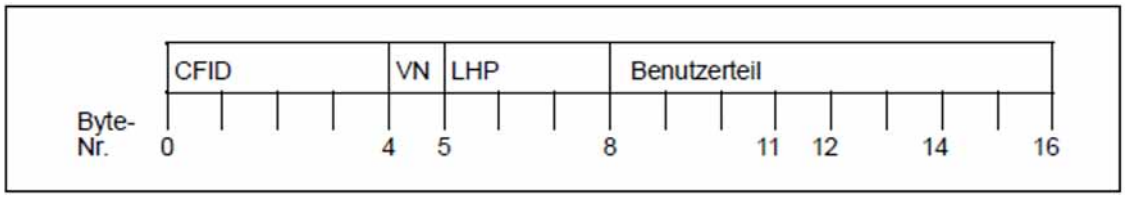
Jeder PAM-Seite einer herkömmlichen K-PAM-Datei ist intern ein außerhalb der Seite gelegener, 16 Byte langer PAM-Schlüssel zugeordnet, in dem DVS-Verwaltungsinformationen geführt werden.

Durch Angabe des Operanden BLKSIZE=(STD,n) im FILE- oder FCB-Makro bzw. des Operanden BUFFER-LENGTH=*STD(n) im ADD-FILE-LINK-Kommando können mehrere PAM-Seiten zu einem Datenblock zusammengefasst werden. Die Grundlage der Verarbeitung mit UPAM bildet jedoch auch bei einer solchen Datei die PAM-Seite, d.h. es werden immer einzelne PAM-Seiten gelesen oder geschrieben.

In einer K-PAM-Datei, bei der mit dem Operanden BLKSIZE=(STD,n) bzw. BUFFER-LENGTH=*STD(n) n PAM-Seiten zu einem Datenblock zusammengefügt wurden, hat ein solcher Datenblock folgenden Aufbau:



Die n * 2048 Byte des Datenblockes stehen gänzlich für Benutzerdaten zur Verfügung, weil die DVS-Verwaltungsinformation in die 16 Byte langen PAM-Schlüssel ausgelagert ist. Der PAM-Schlüssel hat folgenden Aufbau:



Dabei bedeutet:

CFID	Coded File Identification: Verschlüsselter Dateiname der Datei, zu der die PAM-Seitegehört oder zuletzt gehörte. Durch Vergleich mit einer entsprechenden CFID im Katalogeintrag der aktuellen Datei kann das DVS feststellen, ob die Daten der PAM-Seite zur Datei gehören oder ob diese Seite (für diese Datei) zwar reserviert ist, aber noch keine Daten enthält.
VN	Versionsnummer: Die Versionsnummer einer PAM-Seite wird jedes Mal aktualisiert bzw. vom DVS ausgewertet, wenn die Datei erstellt, geändert, eingerichtet oder gelöscht wird. Sie ermöglicht u.a. die partielle Sicherung einer Datei durch ARCHIVE oder HSMS. Dabei werden nur die PAM-Seiten einer Datei gesichert, die seit der letzten Voll- oder Differenzsicherung geändert wurden.
LHP	Logische Half-Page-Nummer: Sie wird zur Konsistenzprüfung einer Datei verwendet sowie zu ihrer Entzerrung, wenn sie von ARCHIVE oder HSMS aus einer Sicherungwiederhergestellt wird.
Benutzerteil	Dieser Teil des PAM-Schlüssels wird von UPAM nicht genutzt und steht dem Benutzer zur Verfügung. Allerdings ist bei der Verarbeitung von SAM- und ISAM-Dateien mit UPAM zu beachten, dass SAM und ISAM dieses Feld verwenden.

> **i** Um die Umstellung von K-PAM- auf NK-PAM-Dateien zu ermöglichen, wird UPAM-Benutzern empfohlen, den PAM-Schlüssel nicht zu benutzen.

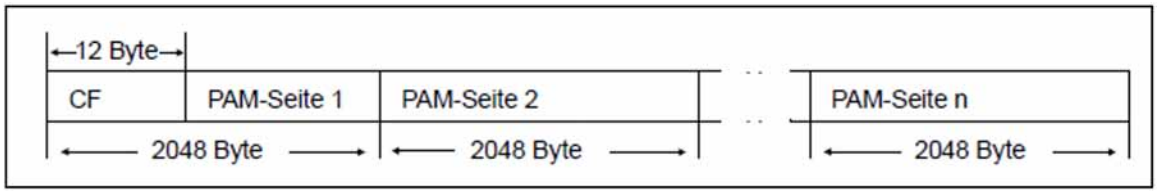
23.1.2 Logischer Block einer NK-PAM-Datei

Im Unterschied zu K-PAM-Dateien besitzen NK-PAM-Dateien außerhalb der PAM-Seiten keine gesonderten PAM-Schlüssel. Je nachdem, welches Blockformat für eine NK-PAM Datei vereinbart ist, werden block-spezifische Verwaltungsinformationen entweder in den logischen Blöcken selbst abgelegt oder überhaupt nicht geführt:

Blockformat „DATA“

Dieses Blockformat wird bei Erstellung der Datei mit dem Operanden BLKCTRL=DATA im Makro FILE bzw. mit dem Operanden BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK im Kommando ADD-FILE-LINK erzeugt. Die block-spezifische Verwaltungsinformation wird in einem Teil jedes logischen Blocks der Datei – dem 12 Byte langen so genannten Blockkontrollfeld (Control Field) – untergebracht.

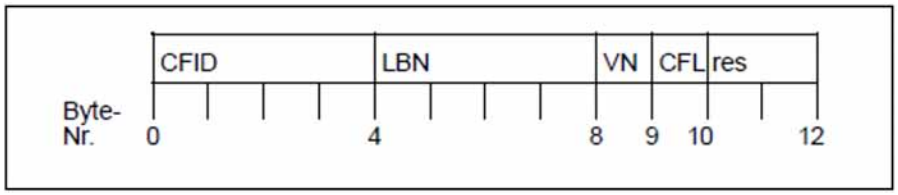
In einer NK-PAM-Datei, bei der mit dem Operanden BLKSIZE=(STD,n) im FILE- oder FCB-Makro bzw. mit dem Operanden BUFFER-LENGTH=*STD(n) im ADD-FILE-LINK-Kommando n 2-KB-Blöcke (Standardblöcke) zu einem logischen Block zusammengefügt wurden, ergibt sich demnach folgender Aufbau für einen solchen logischen Block:



Dabei bedeutet:

CF Blockkontrollfeld (Control Field), 12 Byte lang

Das Blockkontrollfeld hat folgenden Aufbau:



Dabei bedeutet:

CFID Coded File IDentification: Gleiche Bedeutung wie im PAM-Schlüssel (siehe vorangegangenen Abschnitt, "Datenblock einer K-PAM-Datei").

LBN Logical Block Number (Logische Blocknummer): Nummer des Standardblockes, der das Blockkontrollfeld enthält. (Standardblöcke werden innerhalb der Datei fortlaufend nummeriert).

VN Versionsnummer: Gleiche Bedeutung wie im PAM-Schlüssel.

CFL Control FLag: Das Control Flag kennzeichnet Lücken in einer Datei. Diese Information wird vor allem beim Kopieren einer Datei vom bzw. auf Band ausgewertet.

res Für künftige Erweiterungen reservierte Byte des Blockkontrollfeldes.

Der vom 12 Byte langen Blockkontrollfeld beanspruchte Teil eines logischen Blockes steht dem Benutzer nicht mehr für seine Daten zur Verfügung.

Die Länge des für Benutzerdaten nutzbaren Teils eines logischen Blockes aus n Standardblöcken ergibt sich demnach aus folgender Formel:

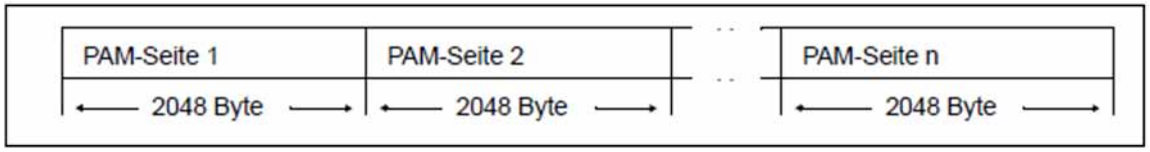
nutzbare Blocklänge:

```
maximale Datenlänge = n * 2048 – 12
```

Blockformat „NO“

Dieses Blockformat wird bei Erstellung der Datei mit dem Operanden BLKCTRL=NO im Makro FILE bzw. mit dem Operanden BLOCK-CONTROL-INFO=*NO im Kommando ADD-FILE-LINK erzeugt. Es werden keine block-spezifischen Verwaltungsinformationen angelegt. Dem Benutzer steht der gesamte logische Block für Nutzdaten zur Verfügung.

In einer NK-PAM-Datei, bei der mit dem Operanden BLKSIZE=(STD,n) im FILE- oder FCB-Makro bzw. mit dem Operanden BUFFER-LENGTH=*STD(n) im ADD-FILE-LINK-Kommando n 2K-Blöcke (Standardblöcke) zu einem logischen Block zusammengefügt wurden, ergibt sich demnach folgender Aufbau für einen solchen logischen Block:



Da der gesamte logische Block für Benutzerdaten zur Verfügung steht, ergibt sich die nutzbare Blocklänge zu n * 2048 Byte (für einen logischen Block, der sich aus n Standardblöcken zusammensetzt).

Das Blockformat „NO“ eignet sich vor allem für Dateien, die auf eine ununterbrochene Folge von Daten – über logische Blockgrenzen hinweg – angewiesen sind (z.B. Lademodule). Systemfunktionen, die block-spezifische Verwaltungsinformationen aus dem PAM-Schlüssel bzw. dem Blockkontrollfeld nutzen, stehen für solche Dateien nicht zur Verfügung (z.B. partielle Sicherung mit ARCHIVE oder HSMS).

23.2 UPAM-Dateiformate

UPAM arbeitet blockorientiert. Grundlage der Verarbeitung ist bei K-PAM-Dateien der 2-KB-Standardblock, bei NK-PAM-Dateien der logische Block, dessen Größe durch den Operanden BLKSIZE im FCB und FILE-Makro bzw. durch den Operanden BUFFER-LENGTH im Kommando ADD-FILE-LINK festgelegt wird.
UPAM kann bis zu 16 2-KB-Standardblöcke gleichzeitig einlesen bzw. ausgegeben. Diese Anzahl wird mit dem Operanden LEN im Makro PAM vereinbart.
Bei der Angabe LEN=(STD,n) bzw. LEN=n*2048 gilt: n <= 16.

Für K-PAM-Dateien gilt:

> Ist der Wert des Operanden LEN im PAM-Makroaufruf kein ganzzahliges Vielfaches von 2048, wird auf das nächstgrößere ganzzahlige Vielfache von 2048 aufgerundet.Bei einer Schreiboperation ist dann der Rest der letzten zu schreibenden PAM-Seite in der Datei undefiniert. Erfolgte dieses Schreiben am Ende der Datei und wird die Datei dann geschlossen, wird die Position des letzten gültigen Bytes dieser PAM-Seite im Last Byte Pointer des Katalogeintrages gespeichert.
> Bei einer Leseoperation wird der Rest der letzten zu lesenden PAM-Seite nicht in den Puffer übertragen.

Für NK-PAM-Dateien gilt:

> Ist der Wert des Operanden LEN im PAM-Makroaufruf kein ganzzahliges Vielfaches der Größe eines logischen Blockes, wird auf das nächstgrößere ganzzahlige Vielfache der logischen Blockgröße aufgerundet. Bei einer Schreiboperation ist dann der Rest des logischen Blockes in der Datei undefiniert. Erfolgte dieses Schreiben am Ende der Datei und wird die Datei dann geschlossen, wird die Position des letzten gültigen Bytes des letzten logischen Blocks im Last Byte Pointer des Katalogeintrages gespeichert.
> Bei einer Leseoperation wird der Rest des letzten zu lesenden logischen Blockes nicht in den Puffer übertragen und der restliche Pufferinhalt ist undefiniert.

Ein Last Byte Pointer mit dem Wert 0 bedeutet, dass der letzte logische Block der Datei komplett gültig ist (Ausnahme siehe Kommando REPAIR-DISK-FILE bzw. Makro VERIF).

Das Kommando SHOW-FILE-ATTRIBUTES gibt den Wert des Last Byte Pointers im Al-location-Teil als LAST-BYTE bzw. in S-Variablen aus, wenn das entsprechende Valid-Bit im Katalogeintrag der Datei gesetzt ist. Der Last Byte Pointer zeigt auf das letzte gültige Byte im letzten logischen Block der Datei.

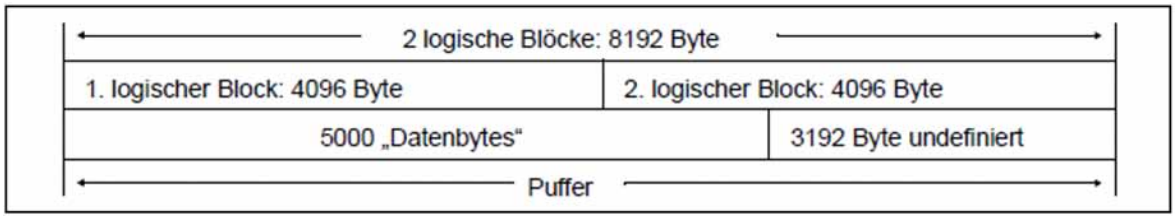
> ¡ Falls Probleme auftreten, kann der Service ggf. den Last Byte Pointer zurücksetzen.

Voraussetzung für die Nutzung des Last Byte Pointers bei PAM-Dateien auf Public-Space ist, dass beim OPEN der Indikator LBP_REQUIRED im Dateisteuerblock (FCB) gesetzt wurde. Mit LBP_REQUIRED wird beim OPEN der LBP im FCB zur Verfügung gestellt und bei CLOSE der aktualisierte Wert im Katalogeintrag gespeichert.

Bei PAM-Node-Files wird stets ein Last Byte Pointer versorgt, unabhängig vom Flag LBP_REQUIRED. Allerdings ist hier zu beachten, dass der LBP im CE veraltet sein kann, da die Datei von einem fremden System geändert worden sein könnte. Deshalb wird im Rahmen des OPEN der aktuell gültige Last Byte Pointer ermittelt und der Anwendung im Dateisteuerblock (FCB) zur Verfügung gestellt.

Beispiel

Für eine Datei mit BLKCTRL=NO und BLKSIZE=(STD,2) (entspricht den Angaben BLOCK-CONTROL-INFO=*NO und BUFFER-LENGTH=*STD(SIZE=2) im Kommando ADD-FILE-LINK) werden in einem PAM-Aufruf die Operanden WRT und LEN=5000 angegeben. Aus dem Puffer werden 5000 Byte übernommen, der Rest bis zum nächstgrößeren ganzzahligen Vielfachen der logischen Blockgröße (8192 Byte) ist undefiniert.



Erfolgte das Schreiben im Beispiel am Ende der Datei und wurde die Datei dann geschlossen, hat der Last Byte Pointer den Wert 904. Damit wurden 904 Bytes im 4. und gleichzeitig letzten logischen Block der Datei geschrieben. Der Last Byte Pointer wird ebenso wie der Last Byte Pointer (hier ist LPP=8) beim Schließen der Datei im Katalogeintrag vermerkt und dient der Anwendung dazu, nach erneutem Öffnen der Datei das genaue Dateiende zu ermitteln.

23.2.1 Umstellung von K-PAM-Dateien auf NK-PAM-Dateien

K-PAM-Dateien können mit dem Kommando COPY-FILE bzw. dem Makro COPFILE in NK-PAM-Dateien mit dem Blockformat „NO“ umgesetzt werden (siehe Beispiel unten), sofern der Benutzerteil des PAM-Keys ungenutzt ist, d. h. den Wert XL8'00' hat. Die Umstellung auf NK-PAM-Dateien mit dem Blockformat „DATA“ kann z.B mit dem Software-Produkt PERCON (siehe Handbuch „PERCON“ [12]) durchgeführt werden.

Beispiel: Umsetzung einer K-PAM-Datei in eine NK-PAM-Datei (Blockformat „NO“) mit dem Kommando COPY-FILE

```
/ADD-FILE-LINK LINK-NAME=DMCOPY22,FILE-NAME=nk-pam-datei,
                BLOCK-CONTROL-INFO=*NO
/COPY-FILE FROM-FILE=k-pam-datei,TO-FILE=nk-pam-datei,
                BLOCK-CONTROL-INFO=*IGNORE-ATTRIBUTE
```

K-PAM-Dateien können auch mit dem Dienstprogramm PAMCONV oder durch Sichern und Restaurieren mit HSMS /ARCHIVE in NK-Dateien mit dem Blockformat „NO“ umgesetzt werden.

Wenn bestehende Anwendungen den PAM-Key nicht benötigen, empfiehlt sich eine Umstellung auf NK-PAM-Dateien auf NK-Platten, damit die bessere Performance und die höhere Ausnutzung des Plattenplatzes genutzt werden können.

23.2.2 Umstellung von NK-PAM-Dateien auf K-PAM-Dateien

NK-PAM-Dateien können mit dem Dienstprogramm PAMCONV (siehe Handbuch „Dienstprogramme“ [13]) oder dem Software-Produkt PERCON (siehe Handbuch „PERCON“ [12]) in K-PAM-Dateien umgesetzt werden.

23.2.3 Kompatibilität des Dateiformats

NK-PAM-Dateien bieten dem Benutzer bei der Verarbeitung den gleichen Funktionsumfang wie K-PAM-Dateien.

Allerdings steht in NK-PAM-Dateien kein dem Benutzerteil eines PAM-Schlüssels vergleichbares Feld zur Verfügung. Daher müssen Anwendungen umgestellt werden, die bisher den Benutzerteil des PAM-Schlüssels in K-PAM-Dateien verwendet haben und nun mit NK-PAM-Dateien arbeiten sollen.

Für NK-PAM-Dateien vom Blockformat „DATA“ gilt folgende *Parallelitätseinschränkung*:
Bei der Verarbeitung einer PAM-Datei mit BLKCTRL=DATA dürfen die IOAREAs nicht parallel für mehrere I/O-Aufträge verwendet werden, weil dabei der Inhalt des Blockkontrollfeldes undefiniert wäre.

Beispiel

- Unzulässig ist:

```
PAM     WRT,FCB1,LOC=BUFFER
PAM     WRT,FCB2,LOC=BUFFER          (Parallele I/Os auf BUFFER)
PAM     WT,FCB1
PAM     WT,FCB2
```

- Zulässig ist:

```
PAM     WRTWT,FCB1,LOC=BUFFER        (Nacheinander ausgeführte I/Os auf BUFFER)
PAM     WRTWT,FCB2,LOC=BUFFER
```

Bei NK-PAM-Dateien vom Blockformat „DATA“ ist die gegenüber K-PAM-Dateien verringerte nutzbare Blocklänge zu beachten: In einem logischen Block aus n Standardblöcken ist die Größe des für Benutzerdaten verfügbaren Teils n * 2048 – 12 Byte.

NK-PAM-Dateien vom Blockformat „NO“ enthalten kein Blockkontrollfeld und damit auch keine Informationen, die denen im Systemteil des PAM-Schlüssels einer K-PAM-Datei entsprechen. Solche Dateien können deshalb auch nicht von ARCHIVE oder HSMS partiell gesichert werden, und bei ihrer Wiederherstellung mit dem Kommando REPAIR-DISK-FILE bzw. dem Makro VERIF müssen sie (ohne Lückentest) bis zum Dateiende, abgerundet auf ein Vielfaches der Blockgröße, bearbeitet werden. Praktisch wirkt sich das dadurch aus, dass Last Page Pointer und ggf. Last Byte Pointer auf Dateiende gesetzt werden.

In einer NK-PAM-Datei mit dem Blockformat „NO“ stehen – wie bei K-PAM-Dateien – die logischen Blöcke in ihrer gesamten Größe für Benutzerdaten zur Verfügung. Im Unterschied zu K-PAM-Dateien kann jedoch bei der Verarbeitung einer solchen Datei nicht mehr auf jeden einzelnen 2-KB-Standardblock eines logischen Blocks zugegriffen werden, sondern nur auf den gesamten logischen Block.

23.3 PAM-Datei eröffnen (OPEN-Modi)

In der folgenden Tabelle sind die OPEN-Modi zusammengestellt.

Open-Modus	Kurzbeschreibung
INPUT	Lesen von Blöcken aus einer vorhandenen Datei
OUTIN	Erstellen einer neuen Datei und ggf. Lesen von Blöcken aus dieser Datei
INOUT	Lesen von Blöcken aus einer vorhandenen Datei und ggf. Hinzufügen und/oder Austauschen von Blöcken

Tabelle 70: Open-Modi bei Eröffnen einer UPAM-Datei

Die folgende Tabelle gibt Auskunft darüber, welche Funktionen des Makros PAM in Verbindung mit den o.g. Open-Modi zulässig sind.

PAM-Makro-Funktionen	OPEN-Modus		
	INPUT	OUTIN	INOUT
RD, RDWT, RDEQU, LRD, LRDWT	X	X	X
WRT, WRTWT, WRTWU	-	X	X
WT, CHK, SYNC	X	X	X
LOCK, UNLOCK, SETL	X	X	X
SETLPP	-	X	X

Tabelle 71: UPAM-Funktionen und zulässige OPEN-Modi

23.4 UPAM-Verarbeitung von Plattendateien

Für Plattendateien bietet UPAM folgende Funktionen:

Erstellen von Plattendateien

Den Zugriff auf Sätze muss der Benutzer selbst programmieren (z.B. sequenziellen Zugriff oder assoziativen Zugriff mittels Hashverfahren).

Lesen und Übertragen von SAM- und ISAM-Dateien

(OPEN INPUT) Lesen von SAM- und ISAM-Dateien und Übertragen auf andere Datenträger (z.B. von Platte auf Band): die Eigenschaften der Dateien werden jeweils im FCB abgesetzt (z.B. BLKSIZE, RECSIZE, RECFORM) bzw. im Kommando ADD-FILE-LINK (BUFFER-LENGTH, RECORD-SIZE, RECORD-FORMAT). Dies ermöglicht es dem Benutzer, den Zugriff auf Sätze zu programmieren.

UPAM kann SAM- oder ISAM-Dateien nicht im UPDATE-Modus eröffnen.

ISAM-Dateien lassen sich wegen der komplexen Zusammenhänge zwischen Index- und Datenblöcken mit UPAM nicht effektiv verarbeiten. Man kann UPAM jedoch dazu benutzen, eine ISAM-Datei blockweise auf ein Band zu übertragen.

Gekettete Ein-/Ausgabe

Bis zu 255 logisch aufeinander folgende PAM-Seiten einer Datei können mit einem PAM-Makroaufruf ein- bzw. ausgegeben werden. Der maximale Wert hängt von den Platteneigenschaften ab (siehe SHOW-MASTER-CATALOG-ENTRY, IO-Länge).

PAM-Makroaufrufe in Listenform

Bis zu 255 PAM-Makroaufrufe (die Aufrufe brauchen sich nicht alle auf die gleiche Datei zu beziehen) können mit einer UPAM-Ein-/Ausgabe-Anforderung abgearbeitet werden, d.h. es wird nur ein SVC benötigt. Die Kettung von PAM-Makros dient (ebenso wie gekettete Ein-/Ausgabe) der zeitlichen Optimierung von Benutzerprogrammen.

Benachrichtigung des Benutzerauftrags

Bei Beendigung einer UPAM-Ein-/Ausgabeoperation und Start eines Contingency-Prozesses (Eventing-Mechanismus) wird der Benutzerauftrag benachrichtigt.

DRV

Bei Dateien mit Dual Recording by Volume (DRV; siehe Handbuch „DRV“ [14]) kann sich der Benutzer über den aktuellen Zustand (gegebenenfalls Kopien-Ausfall) informieren. Die Information wird bei der Ausführung einer Ein-/Ausgabe von UPAM angefordert und im FCB (Feld ID1DRVST) hinterlegt. Dieses Feld wird jedoch nur bei einer Änderung des DRV-Status aktualisiert.

Shared-Update-Verarbeitung (Multi-User-Betrieb)

Mehrere parallele Aufträge können eine PAM-Datei gleichzeitig bearbeiten.

Plattendatei ohne PAM-Schlüssel

Eine Datei ohne PAM-Schlüssel wird im Makro FILE oder FCB mit den Operanden BLKCTRL=NO / DATA bzw. im Kommando ADD-FILE-LINK mit den Operanden BLOCK-CONTROL-INFO=*NO/WITHIN-DATA-BLOCK vereinbart. Dabei ist auf Folgendes zu achten:

- Die Datei muss Standardblöcke haben (BLKSIZE=(STD,n)) bzw. BUFFER-LENGTH=*STD(SIZE=n)

- Falls es sich nicht um eine ISAM-Datei handelt (FCBTYPE=SAM/PAM bzw. ACCESS-METHOD=*SAM/*UPAM), muss die Sekundärzuweisung mindestens so groß sein wie die vereinbarte Blockgröße (BLKSIZE bzw. BUFFER-LENGHTH).

Für die Zugriffsmethode UPAM gilt

- Bei SHARUPD=*WEAK kann eine Datei auch dann lesend eröffnet werden, wenn sie von einer anderen Anlage der Multi-System-Umgebung schreibend eröffnet ist. Allerdings wird eine Dateierweiterung (Sekundärzuweisung), die von der anderen Anlage ausgelöst wird und zeitlich *nach* dem lokalen Eröffnen liegt, nicht zur Kenntnis genommen. Diese später erzeugten Dateibereiche können also vom lokalen Benutzer nicht gelesen werden.
- Nicht erlaubte Kombinationen führen zu einem OPEN-Fehler.
- Soll eine PAM-Datei von verschiedenen Aufträgen gleichzeitig bearbeitet werden, muss für jeden Auftrag im Makro FCB der Operand SHARUPD=YES bzw. im Kommando ADD-FILE-LINK der Operand SHARED-UPDATE=*YES angegeben werden.

  > **i** Bei UPAM gibt es keine implizite Sperre oder Freigabe von PAM-Seiten. Der Benutzer muss jede Sperre selbst anfordern bzw. freigeben. Die einzige Ausnahme ist der CLOSE-Makroaufruf: bevor die Datei geschlossen wird, werden alle PAM-Seiten freigegeben, die dieser Auftrag in der Datei noch gesperrt hat.

- Ist eine PAM-Seite gesperrt, kann ein anderer Auftrag dennoch an ihm Lese- bzw. Schreiboperationen ausführen. Verhindert wird nur, dass ein anderer Auftrag diese PAM-Seite noch einmal sperrt. Deshalb sollten alle Aufträge bei Simultan-Aktualisierung nach folgendem Schema vorgehen:
  `Sperren (LOCK) -> Lesen (RD) -> Schreiben (WRT) -> Freigeben (UNLOCK)`

  Die Angaben in Klammern bezeichnen den Operanden im PAM-Makroaufruf.

  Der Benutzer sollte nicht versuchen, andere Betriebsmittel exklusiv anzufordern, während er PAM-Seiten gesperrt hat, da dies zu einer Deadlock-Situation führen kann (EXLST-Ausgang DLOCK).
- Ein Auftrag kann nicht mehr als 255 PAM-Seiten gleichzeitig sperren.
- Ein und dieselbe PAM-Seite kann gleichzeitig nur einmal innerhalb einer UPAM-Operandenlistenkette gesperrt werden.
- UPAM wertet das FCB-Feld ID1TOUT (Wert des PAMTOUT-Operanden) aus, wenn eine Sperre angefordert wird.
  Daher kann das Benutzerprogramm diesen Wert beliebig verändern, solange die Datei geöffnet ist. Werden bei Kettung von PAM-Makroaufrufen mehrere Sperren angefordert, gilt der PAMTOUT-Wert des FCB, auf den sich die erste Sperre bezieht, für alle in dieser „Kette“ angeforderten Sperren.
- Sind Sperrungen nicht unmittelbar verfügbar, wird der Auftrag für die im Feld ID1TOUT angegebene Zeit in eine Warteschlange gebracht. Ist nach dieser Zeit die angeforderte Sperre immer noch nicht verfügbar, wird entweder der DLOCK- oder der PGLOCK-Ausgang gewählt, abhängig davon, ob der Auftrag momentan PAM-Seiten gesperrt hat oder nicht.
- Eine LOCK- bzw. UNLOCK-Operation, angewendet auf eine mit SHARUPD=*NO oder SHARUPD=*WEAK eröffnete Datei wird als Nulloperation behandelt; lediglich der Dateizeiger wird aktualisiert.

- Bei einem PGLOCK kann das Programm normal fortfahren, z.B. nach einer Wartezeit erneut versuchen, den Block zu sperren. Beim DLOCK liegt zumindest der Versuch einer Mehrfachsperrung vor, und das Programm gerät in einen instabilen Zustand. Während dieses instabilen Zustandes löst jeder Versuch, eine PAM-Seite zu sperren, bevor alle aktuellen Sperren freigegeben sind, eine vorzeitige Programmbeendigung aus. Nur wenn in oder nach der Routine für den DLOCK-Ausgang alle aktuellen Sperren freigegeben worden sind, wird das Programm wieder stabil und kann dann wie üblich Sperren anfordern.
- Mit UPAM kann auch auf SAM-Node-Files zugegriffen werden. Für schreibenden Zugriff (OPEN OUTIN) muss zusätzlich das Bit SAM_NODE_FILE_ENABLE im FCB gesetzt werden. Standardmäßig erhält UPAM beim Lesen aus SAM-Node-Files SAM-Blöcke in gleicher Struktur wie von Public-Space. Ebenso muss UPAM beim Schreiben Datenblöcke mit SAM-Struktur schreiben.
  Davon abweichend ist es möglich ohne Konvertierung der Daten vom UFS in das SAM-Block-Format zu arbeiten:
  Der sogenannte RAW-Zugriff ermöglicht den direkten Zugriff in die Daten des Node-Files. Der RAW-Zugriff wird durch Setzen des Indikators UPAM_RAW_ACCESS im FCB vor der OPEN-Verarbeitung aktiviert. Ein dynamisches Umschalten von blockorientierter in RAW-Verarbeitung ist nicht möglich.
  Falls nicht an das Dateiende geschrieben wird, muss in beiden Fällen (blockorientiert mit Konvertierung oder im RAW-Modus ohne Konvertierung) unbedingt darauf geachtet werden, dass die zu schreibenden (Nutz-)Daten exakt genauso lang sind wie die Daten, die überschrieben werden sollen. Ansonsten muss mit Datenverlust gerechnet werden. Beim Zugriff im RAW-Modus muss außerdem beachtet werden, dass UPAM die Daten in Vielfachen von 2kB addressiert und damit in der Regel ohne Beachtung von Satzanfang und -ende bzw. Satzlänge. Für die korrekte Verarbeitung der Daten ist die Anwendung verantwortlich.

23.4.1 Gekettete Ein-/Ausgabe

Gekettete Ein-/Ausgabe ermöglicht die gleichzeitige Ein-/Ausgabe von bis zu 255 logisch aufeinander folgenden PAM-Seiten mit einem PAM-Makroaufruf (nicht zu verwechseln mit der Verkettung von PAM-Makroaufrufen in Listenform mit dem Operanden CHAIN). Sie vermindert so die Anzahl von Ein-/Ausgabe-Operationen (sowie Unterbrechungen) und führt zu einer Zeitersparnis bei der Verarbeitung. Auf der anderen Seite werden jedoch Arbeitsspeicherbedarf und Paging-Aufwand größer.

UPAM arbeitet mit geketteter Ein-/Ausgabe, wenn der Operand LEN im PAM-Makroaufruf einen Wert > STD bzw. > 2048 enthält.

Dateiende-Verarbeitung (EOF-Verarbeitung)

Bei einer Schreiboperation wird eine Sekundärzuweisung durchgeführt; die angegebenen PAM-Seiten werden der Datei angefügt.

Tritt bei einer Leseoperation die Dateiendebedingung auf, überträgt UPAM nur die zur Datei gehörenden PAM-Seiten in den Puffer.

UPAM informiert den aufrufenden Auftrag über die EOF-Verarbeitung folgendermaßen:

- *ohne ereignisgesteuerte Verarbeitung:*
  Der Benutzerauftrag erhält die Steuerung am EXLST-Ausgang USERERR mit dem Fehlercode X'0922' im Feld ID1ECB des FCB. Das Feld ID1NBPP des FCB enthält die Anzahl der übertragenen PAM-Seiten. Ist der Wert dieses Feldes X'00', liegen alle zu lesenden PAM-Seiten außerhalb der Datei. Ist der Wert dieses Feldes größer als X'00', so hat der Benutzerauftrag eine Warte-Operation auszuführen, falls diese nicht in der Lese-Operation implizit enthalten war (d.h. in einer RDWT-Operation).
- *mit ereignisgesteuerter Verarbeitung:*
  Liegen alle zu lesenden PAM-Seiten außerhalb der Datei, übergibt UPAM dem Benutzerauftrag die Steuerung am EXLST-Ausgang USERERR mit dem Fehlercode X'0922' im Feld ID1ECB des FCB. Gehört wenigstens eine der zu lesenden PAM-Seiten zur Datei, setzt UPAM den Basisprozess fort oder startet einen Contingency-Prozess (siehe Abschnitt „TU-Eventing: ereignisgesteuerte Verarbeitung“). Jetzt enthält das Feld IDECBNPA des FECB (= File Event Control Block, siehe "TU-Eventing: ereignisgesteuerte Verarbeitung") eine Anzeige, wie viele PAM-Seiten übertragen worden sind:

X'00'	alle zu lesenden PAM-Seiten wurden in den Puffer übertragen
X'0n'	n= Anzahl der PAM-Seiten, die zur Datei gehören und in den Puffer übertragen wurden

Sperren und Freigeben von PAM-Seiten

Es genügt, die erste einer Reihe zu sperrender/freizugebender PAM-Seiten in einem PAM-Makroaufruf anzugeben; die Anzahl der zu sperrenden/freizugebenden Seiten ergibt sich aus dem Operanden LEN. Es ist jedoch zu beachten, dass der Dateizeiger nach einer LOCK- bzw. UNLOCK-Operation auf die letzte PAM-Seite verweist, die gesperrt/freigegeben wurde. Sie kann außerhalb der Datei liegen (s.o. „Dateiende-Verarbeitung“).

Eine LOCK- bzw. UNLOCK-Operation auf eine SHARUPD=*NO oder SHARUPD=*WEAK eröffnete Datei ist eine Nulloperation: Es wird lediglich der Zeiger auf die zuletzt bearbeitete Seite aktualisiert, dabei wird der LEN-Operand ausgewertet.

Verarbeitung von PAM-Schlüsseln

Für die Verarbeitung von PAM-Schlüsseln gibt es zwei Möglichkeiten:

- Der Benutzer liest/schreibt jeden einzelnen Schlüssel einer Reihe von PAM-Seiten: PAM-Makro-Operand MKEY=YES; Operand KEYFLD muss die Adresse eines genügend großen Bereichs angeben.
- Der Benutzer liest/schreibt nur den ersten Schlüssel einer Reihe von PAM-Seiten. Beim Schreiben wird den folgenden Blöcken derselbe Schlüssel zugeordnet wie dem ersten Block, lediglich die logische Blocknummer wird jeweils um 1 weitergezählt.

Hinweise

Da ein Block erst bei einem expliziten Aktionsmakroaufruf in den Benutzerpuffer übertragen wird, entsteht eine Verzögerung. Deshalb muss eine asynchrone Ein-/Ausgabe mit dem Aktionsmakroaufruf WT beendet werden. Bei TU-Eventing sollte man den Makro SOLSIG (asynchron oder synchron) verwenden.

Bei jedem nicht erfolgreichen Sprung in die UPAM-Routinen wird die Steuerung an die Routine übertragen, die im EXIT-Operanden des FCB angegeben ist bzw. an die entsprechende EXLST-Routine. Im FCB wird ein Kennzeichen gespeichert.

Bei jeder von UPAM veranlassten Programmbeendigung versorgt UPAM die Register 0, 1 und 15, die in Speicherauszügen leicht auszuwerten sind.

Register	Inhalt
0	Adresse, an der der Abbruch auftrat
1	Adresse des Elements einer UPAM-Operandenlistenkette, in dem der Fehler entdeckt wurde
15	UPAM-Fehlercode

Wenn Register 1 beim ersten Aufruf des PAM-Makros eine ungültige Adresse enthält, wird im Speicherauszug das Register 0 diese ungültige Adresse und Register 1 Nullen enthalten (d.h. der Fehler trat auf, bevor das erste Element der Operandenlistenkette gefunden wurde).

UPAM verwendet folgende EXLST-Ausgänge:

EXLST-Ausgang	Kurzbeschreibung
ERRADDR	Hardwarefehler oder abnormale Ein-/Ausgabebeendigung
USERERR	Unzulässige Makroanwendung im Programm oder Lesezugriff auf eine nicht zur Datei gehörende PAM-Seite (Dateiende)
EOFADDR	Versuch, eine *DUMMY-Datei zu lesen
PGLOCK	Nicht alle angeforderten Sperren sind innerhalb der gegebenen Zeit verfügbar und der Auftrag hält momentan keine Sperren
DLOCK	Die Anforderung einer Sperre wird abgewiesen und der Auftrag hält bereits Sperren

PAM-Seiten, die einer Datei zugewiesen, aber vom Eigentümer dieser Datei noch nicht geschrieben wurden, sind an ihrem systemintern verschlüsselten Dateinamen (CFID=Coded File ID, Byte 0-3 des PAM-Schlüssels bzw. des Blockkontrollfeldes) zu erkennen, der dann mit dem aktuellen Dateinamen nicht übereinstimmt. Der Vergleich ist vom Benutzer durchzuführen. Dabei ist darauf zu achten (siehe auch Operand KEYFLD im PAM-Makroaufruf), dass zum OPEN-Zeitpunkt die aktuelle CFID in das erste Wort im Feld ID1KEY1 des FCB geschrieben wird.

Die folgenden Punkte gelten nur bei der Verarbeitung von K-PAM-Dateien (BLKCTRL=PAMKEY bzw. BLOCK-CONTROL-INFO=*PAMKEY):

- Nach Ausführung einer RDWT-, LRDWT- oder RDEQU-Operation steht die CFID des gelesenen Blocks im ersten Wort des FCB-Feldes ID1KEY2.
- Nach Ausführung einer der Operationen WRT, WRTWT, WRTWU, WT steht die CFID der betreffenden PAM-Seite im ersten Wort des FCB-Feldes ID1KEY1. Der von OPEN erstellte Eintrag wird überschrieben; er sollte daher vor der Verarbeitung für spätere Vergleiche sichergestellt werden.
- Nach Ausführung der Operationen LRD und RD ist der Inhalt der Felder ID1KEY1 und ID1KEY2 unverändert.
- Bei ereignisgesteuerter Verarbeitung steht nach dem Abschluss einer Ein-/Ausgabeoperation die CFID der betreffenden PAM-Seite im ersten Wort des FCB-Feldes ID1KEY1.

Die Felder ID1LWB (PARMOD=24) bzw. ID1LWBPT (PARMOD=31) im FCB enthalten die Adresse des letzten Blocks, auf dem von UPAM eine Ein-/Ausgabeoperation erfolgreich durchgeführt wurde. Als Anzeige für einen evtl. noch ausstehenden WT wird das linke Byte des Feldes ID1LWB verwendet.

War die letzte UPAM-Operation für die Datei ein erfolgreicher WT, erhält die Anzeige in ID1LWB den Wert X'00' und die drei niederwertigen Byte des Feldes ID1LWB bzw. das Feld ID1LWBPT enthalten die Adresse des Blocks, auf den der WT sich bezog. Dabei spielt es keine Rolle, ob die den WT auslösende Operation erfolgreich abgeschlossen wurde.

Hat die letzte UPAM-Operation für die Datei keinen WT ausgelöst, wird die Anzeige in ID1LWB auf den Wert X'FF' gesetzt. Der Inhalt der rechten drei Byte von ID1LWB bzw. ID1LWBPT ist dann ohne Bedeutung.

23.5 UPAM-Verarbeitung von Banddateien

Für Banddateien bietet UPAM folgende Funktionen:

Erstellen von Banddateien

UPAM erstellt Banddateien, die nicht über ein Band hinausgehen. Den Zugriff auf logische Sätze dieser Dateien muss der Benutzer selbst programmieren.

Lesen von SAM-Dateien mit Standardblöcken

Die Dateieigenschaften werden durch die OPEN-Verarbeitung (siehe Abschnitt „Ablauf der OPEN-Verarbeitung“) im FCB abgesetzt, z.B. BLKSIZE, RECSIZE, REC-FORM bzw. durch die Operanden BUFFER-LENGTH, RECORD-SIZE, RECORD-FORMAT im Kommando ADD-FILE-LINK. Dies ermöglicht es dem Benutzer, Satzzugriff zu programmieren.

Gekettete Ein-/Ausgabe

Sie ist bei Banddateien nicht möglich!

Benachrichtigung des Benutzerauftrags

Bei Beendigung einer UPAM-Ein-/Ausgabeoperation und Start eines Contingency-Prozesses (Eventing-Mechanismus) wird der Benutzerauftrag benachrichtigt.

Hinweise

Bei der Anwendung von UPAM auf eine Banddatei ist Folgendes zu beachten:

- Die Datei muss auf einem einzigen Band Platz finden.
- Jede Schreib-/Leseoperation bearbeitet genau einen physikalischen Block.
- Bei einer Datei vom Format BLKCTRL=PAMKEY bzw. BLOCK-CONTROL-INFO=*PAMKEY (explizit oder implizit) muss dies ein Standardblock der Länge 2064 Byte sein: In den ersten 16 Byte steht der PAM-Schlüssel, die restlichen 2048 Byte enthalten die Benutzerdaten. Der Operand LEN des PAM-Makros darf nur die Werte STD oder 2048 annehmen.
- Bei einer Datei vom Format BLKCTRL=DATA / NO (entspricht BLOCK-CONTROL-INFO=*WITHIN-DATA-BLOCK / *NO im Kommando ADD-FILE-LINK) kann man den Wert für BLKSIZE (bzw. BUFFER-LENGTH) in Byte angeben, wobei der Wert ein Vielfaches von 2048 sein muss. Der Wert für LEN im Makro PAM darf den für BLKSIZE nicht übersteigen. Auf das Band werden Blöcke der Größe LEN geschrieben, die entweder nur Benutzerdaten (bei BLKCTRL=NO) oder 12 Byte Blockkontrollfeld und (LEN – 12) Byte Benutzerdaten (bei BLKCTRL=DATA) enthalten. Diese Blöcke können mit der UPAM-Funktion RD gelesen werden, wobei für den Wert LEN_{RD} von LEN im Leseaufruf gilt:

  $$LEN_{WR} <= LEN_{RD} <= BLKSIZE$$

  (LEN_{WR}: Wert von LEN beim Schreiben der Datei)

  Bei einer Datei mit FCBTYPE=ISAM bzw. ACCESS-METHOD=*ISAM ist in diesem Zusammenhang folgende Besonderheit zu beachten: Unabhängig von der BLKSIZE-Angabe arbeitet UPAM stets mit der Blockgröße 2048 Byte, da bei ISAM jeder 2K-Block ein Blockkontrollfeld besitzt und daher eine eigene Einheit darstellt. In diesem Fall darf der Wert für LEN 2048 nicht übersteigen.

- Bei UPAM-Zugriff auf eine Banddatei mit BLKCTRL=PAMKEY und Blöcken, die vom Standardformat (2064 Byte) abweichen, können folgende Fehler auftreten:
  - Hardware-Fehler (Fehlercode = 927);
  - die Daten werden im Puffer falsch abgespeichert;
  - der PAM-Schlüssel wird verfälscht.
- Auf Banddateien darf nicht von mehreren Aufträgen, bzw. von einem Auftrag mehrfach zugegriffen werden (was auf die Eigenschaften der Magnetbänder zurückzuführen ist); das heißt:
  - eine Banddatei kann nicht mit SHARUPD=YES oder WEAK eröffnet werden;
  - eine Banddatei, die bereits eröffnet ist, kann nicht nochmals eröffnet werden, auch wenn beide Eröffnungen vom Typ INPUT sind!
  - eine Banddatei kann nicht mit einem FCB-Operanden PAMREQS > 1 eröffnet werden.
- Es ist möglich, mit UPAM eine Banddatei im wahlfreien Zugriff zu lesen. Der Zeitaufwand hierfür kann aber beträchtlich sein.
- Es ist möglich, ab einer bestimmten Stelle einer bestehenden Banddatei PAM-Blöcke zu schreiben. Der letzte neu geschriebene Block wird automatisch zum letzten Block der Datei, auch wenn die bestehende Datei mehr Blöcke enthielt. Die Datei kann dann nur noch bis zu diesem Block gelesen werden. Wird anschließend ein (weiter vorne liegender) Block der Datei gelesen und die Datei daraufhin geschlossen, so wird der zuletzt gelesene Block zum letzten Block der Banddatei.

Hinweise zur Programmierung

Bei jeder von UPAM veranlassten Programmbeendigung versorgt UPAM die Register 0, 1 und 15, die in Speicherauszügen leicht auszuwerten sind.

Register	Inhalt
0	Adresse, an der der Abbruch auftrat
1	Adresse des Elements einer UPAM-Operandenlistenkette, in dem der Fehler entdeckt wurde
15	UPAM-Fehlercode

Enthält Register 1 nach dem ersten PAM-Makroaufruf eine ungültige Adresse, ist diese Adresse im Speicherauszug im Register 0 zu finden; das Register 1 hat dann den Wert 0. D.h., der Fehler trat auf, bevor das erste Element der Operandenlistenkette gefunden wurde.

UPAM verwendet folgende EXLST-Ausgänge:

EXLST-Ausgang	Kurzbeschreibung
ERRADDR	Hardwarefehler oder abnormale Ein-/Ausgabebeendigung
USERERR	Unzulässige Makroanwendung im Programm oder Lesezugriff auf eine nicht zur Datei gehörende PAM-Seite (Dateiende)

Die Felder ID1LWB (PARMOD=24) bzw. ID1LWBPT (PARMOD=31) im FCB enthalten die Adresse des letzten Blocks, auf dem von UPAM eine WT-Operation erfolgreich durchgeführt wurde. Als Anzeige wird das linke Byte des Feldes ID1LWB verwendet.

War die letzte UPAM-Operation für die Datei ein erfolgreicher WT, erhält die Anzeige in ID1LWB den Wert X'00' und die drei niederwertigen Byte des Feldes ID1LWB bzw. das Feld ID1LWBPT enthalten die Adresse des Blocks, auf den der WT sich bezog. Dabei spielt es keine Rolle, ob die den WT auslösende Operation erfolgreich abgeschlossen wurde.

Hat die letzte UPAM-Operation für die Datei keinen WT ausgelöst, wird die Anzeige in ID1LWB auf den Wert X'FF' gesetzt. Der Inhalt der rechten drei Byte von ID1LWB bzw. ID1LWBPT ist dann ohne Bedeutung.

Magnetbandkassetten werden von UPAM wie Magnetbänder behandelt.

23.6 Kettung von PAM-Makroaufrufen in Listenform

PAM-Makroaufrufe, die miteinander verkettet werden sollen, sich aber nicht notwendig auf die gleiche Datei beziehen müssen, sind mit dem Operanden MF=L in Listenform zu erzeugen und in einem Konstantenbereich unterzubringen; die Kettung wird durch Angabe des Operanden CHAIN= erreicht.

Die Makroaufrufe haben (bis auf den Letzten) folgendes Format:

	Operation	**Operanden**
$element_n$	PAM	fcbadr,operation,...,MF=L,CHAIN=$element_{n+1}$

Beim letzten Element der Kette entfällt der Operand CHAIN.

Der Aufruf einer Kette von PAM-Makroaufrufen in Listenform erfolgt mit einem PAM-Makro der folgenden Form:

	Operation	**Operanden**
	PAM	MF=(E,$element_1$)

Es wird jeweils nur ein SVC pro Kette ausgeführt, d.h. man vermeidet durch Kettung von UPAM-Anforderungen den Aufwand einer mehrfachen SVC-Bearbeitung. Die Elemente einer Kette müssen sich nicht notwendig auf die gleiche Datei beziehen.

Beispiel zur Codierung

```
START
        LDBASE 10
        USING *,10
        .
        .
        PAM MF=(E,ELEM1)
        .
        .
        .
        TERM
*KONSTANTENBEREICH
ELEM1   PAM     ......,MF=L,CHAIN=ELEM2
ELEM2   PAM     ......,MF=L,CHAIN=ELEM3
ELEM3   PAM     ......,MF=L
        .
        .
        .
        END
```

Bei fehlerfreiem Ablauf erhält der Benutzer die Kontrolle bei der Anweisung, die auf den PAM-Makroaufruf folgt, der die Abarbeitung einer Operandenlisten-Kette forderte.

Alle Operationen werden in genau der Reihenfolge ausgeführt, in der die PAM-Makro-Listen innerhalb der Kette vorkommen – mit einer Ausnahme: Wenn die erste Operation, die eine Sperre fordert, erkannt wird, wird der Rest der Kette durchgeprüft, und alle Operationen, die Sperren fordern, werden registriert. Falls innerhalb der angegebenen Zeit nicht alle geforderten Sperren verfügbar sind, wird die Kette bei der Operation abgebrochen, die die erste Sperre forderte.

Wird eine in der Kette angeforderte Aktion nicht erfolgreich durchgeführt, werden auch die auf diese Anforderung folgenden Aktionen nicht ausgeführt (einschließlich Sperren). Die Steuerung wird an den entsprechenden EXLST-Ausgang übergeben; das Register 1 weist auf den FCB der fehlerauslösenden Datei. Der Fehler-Code (ID1ECB) und das Fehlerbyte (ID1XITB) werden in diesem FCB wie üblich gesetzt. Das ID1CHERR-Feld im FCB wird auf die Adresse desjenigen Elements der Operandenlisten-Kette gesetzt, in dem der Fehler auftrat.

Eine Prüfoperation auf eine noch nicht abgeschlossene Ein-/Ausgabe-Operation führt ebenfalls dazu, dass die Kontrolle aus der Operandenlisten-Kette an das Benutzerprogramm
an der angegebenen Adresse übergeben wird. Die restlichen Anforderungen in der Kette (einschließlich Sperrungen) werden nicht ausgeführt, und das ID1CHERR-Feld im FCB wird auf die Adresse desjenigen Elements der Operandenliste gesetzt, das den CHK enthält. Es ist also davon abzuraten, Prüfvorgänge in Operandenlisten-Ketten abzusetzen.

Der Benutzer muss sicherstellen, dass innerhalb einer Kette die Operationen Sperren, Lesen, Schreiben, Warten, Prüfen und Freigeben sinnvoll angewendet werden. Puffer und Schlüsselfelder werden von UPAM entsprechend der Anforderung benutzt. Die Existenz eines Puffers und die Zugriffsbefugnis werden überprüft, aber es gibt keine Garantie, dass ein Puffer oder ein Schlüsselfeld, die von einer Operation in einer Kette gefüllt werden, von einer späteren Operation dieser Kette nicht überschrieben werden.

Ein Pseudo-Programmabschnitt (DSECT), der mit dem Makroaufruf IDPPL generiert werden kann, beschreibt das Format der PAM-Operandenliste.

In allen Fällen, in denen eine Kette von UPAM-Operandenlisten nicht vollständig abgearbeitet werden kann (z.B. eine Ein-/Ausgabe-Operation ist gescheitert, ein Fehler wurde entdeckt, eine Sperre konnte nicht ausgeführt wurden, die EOF-Bedingung trat auf oder ein CHK wurde an eine laufende Ein-/Ausgabe-Anforderung gestellt), wird in das ID1CHERR-Feld des FCB die Adresse des ersten nicht ausgeführten Eintrags der Kette gebracht. Der Benutzer kann davon ausgehen, dass alle Einträge davor richtig ausgeführt wurden.

UPAM meldet über FCB-EXIT und EXLST keinen Fehler,

- wenn der UPAM-SVC ausgeführt wird und weder Register 1 noch ein Operand der Kette eine gültige Adresse enthalten (z.B.: die Adresse liegt nicht an einer Wortgrenze oder beschreibt ein Feld, das nicht ganz zum virtuellen Adressraum des Benutzers gehört und nicht groß genug ist, um eine UPAM-Makro-Operandenliste aufzunehmen usw.);
- wenn die FCB-Adresse in der UPAM-Makro-Operandenliste fehlt oder ungültig ist.

In beiden Fällen gibt es keinen FCB, dem der Fehler gemeldet werden könnte, deshalb veranlasst UPAM eine vorzeitige Beendigung des Programms.

Eine UPAM-Operandenlisten-Kette wird vollständig für gültig erklärt, bevor irgendeine Aktion ausgelöst wird. Ist eine CHAIN-Adresse oder eine FCB-Adresse ungültig, wird der Auftrag vorzeitig beendet, bevor irgendein Element der Kette ausgeführt wird.

Wird der Eventing-Mechanismus verwendet und steht bei Ende der Ein-/Ausgabe keine Ereigniskennung zur Verfügung, an die das Ereignis gemeldet werden kann, wird das Benutzerprogramm ebenfalls abgebrochen.

23.7 TU-Eventing: ereignisgesteuerte Verarbeitung

Die im Folgenden beschriebene ereignisgesteuerte Verarbeitung sowie die genannten Makroaufrufe sind ausführlich im Handbuch „Makroaufrufe an den Ablaufteil“ [2] dargestellt.

Ereignisgesteuerte Verarbeitung wird von UPAM dazu benutzt, einem Auftrag die Beendigung einer angeforderten Ein-/Ausgabe mitzuteilen. Der Auftrag kann:

- parallel zur UPAM-Ein-/Ausgabe fortgesetzt werden und bei Eintreten des erwarteten Ereignisses (hier die Beendigung der angeforderten Ein-/Ausgabe) mit einem Contingency-Prozess fortfahren (asynchrone Verarbeitung).
- auf die Beendigung der angeforderten Ein-/Ausgabe warten und dann fortfahren (synchrone Verarbeitung, die natürlich auch ohne ereignisgesteuerte Verarbeitung möglich ist).

UPAM gibt nach Vollendung einer Ein-/Ausgabe-Operation eine Meldung an die zugeordnete Ereigniskennung (mit dem Makroaufruf POSSIG). Diese Meldung trifft dann sofort oder später auf die vom Benutzer ausgegebene Anforderung (SOLSIG-Makroaufruf). Liegen sowohl Anforderung als auch Meldung vor (für dieselbe Ereigniskennung), wird ein Contingency-Prozess gestartet oder der Basisprozess fortgesetzt.

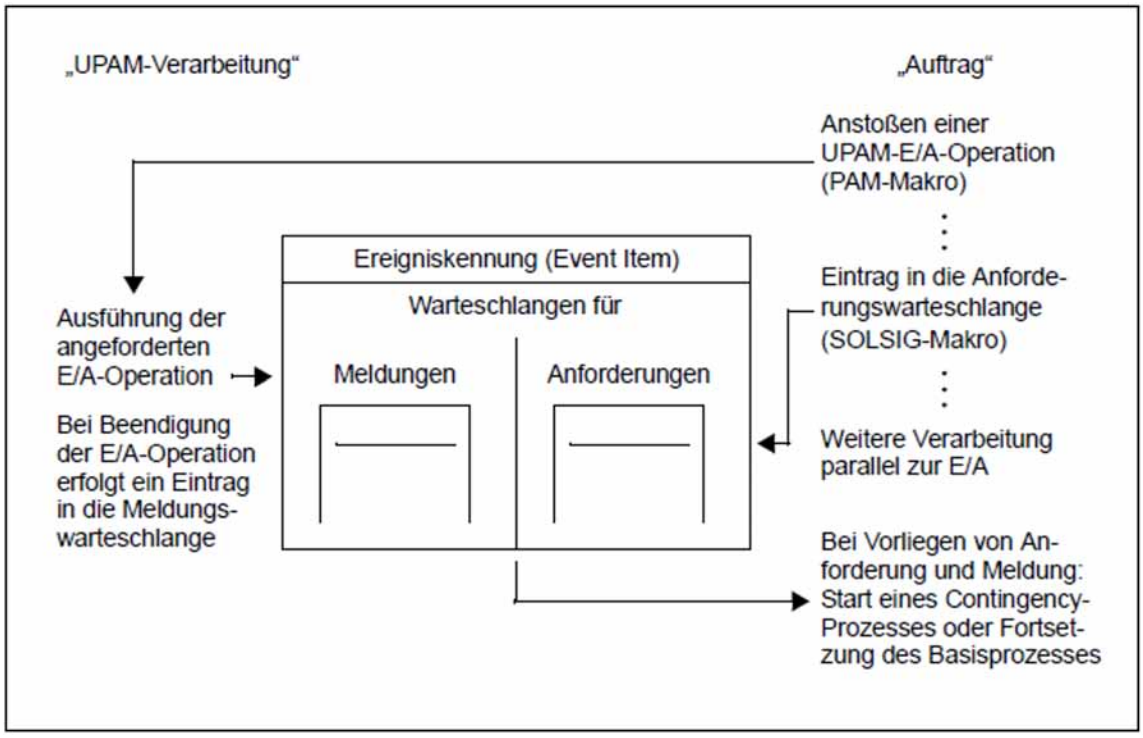


Bild 40: Koordinierung von Benutzerauftrag und UPAM-Verarbeitung

Basisprozess

Dem System müssen die zu verwendenden Ereigniskennungen und Contingency-Definitionen mitgeteilt werden (Makroaufrufe ENAEI, ENACO).

Für jede Ein-/Ausgabe muss dem System die Adresse eines FECBs (File Event Control Block) mitgegeben werden. Parallel laufende Ein-/Ausgaben müssen auf verschiedene FECBs verweisen. Die maximale Anzahl paralleler Ein-/Ausgaben wird durch den FCB-Operanden PAMREQS festgelegt; für Banddateien gilt: PAMREQS=1

Die Zahl der Contingency-Definitionen richtet sich danach, ob unterschiedliches Vorgehen nach einer ausgeführten Ein-/Ausgabe-Operation gewünscht ist. Soll z.B. immer auf die gleiche Weise verfahren werden, genügt es, die Contingency-Definition nur einmal zu codieren.

Für jede Ereigniskennung muss ein 14 Byte langer Steuerblock eingerichtet werden: der FECB = File Event Control Block.

Solange die erste Ein-/Ausgabe-Operation mit einem FECB nicht beendet ist, darf dieser FECB nicht für andere Ein-/Ausgabe-Operationen benutzt werden.

Bei jeder UPAM-Ein-/Ausgabe-Anforderung muss die Adresse des zugeordneten Steuerblocks angegeben werden (Operand FECB= im PAM-Makroaufruf). Es dürfen durch den PAM-Makroaufruf keine Warteoperationen angefordert werden, weder explizit noch implizit. Die Befehlsfolge Lesen (RD) -> Schreiben (WRT) -> Warten (WT) auf denselben Block führt demnach zu einem undefinierten Ergebnis. Das Ereignis (Beendigung der Ein-/Ausgabe) muss vor dem WRITE-Aufruf abgewartet werden.

Nach jeder UPAM-Ein-/Ausgabe-Anforderung muss genau eine Anforderung an die zugeordnete Ereigniskennung abgegeben werden (Makroaufruf SOLSIG). Mit dieser Anforderung kann festgelegt werden, ob der Basisprozess parallel zur Ein-/Ausgabe weiterläuft oder deren Beendigung abwartet.

Beim Start eines Contingency-Prozesses werden diesem in den Registern 2 und 3 folgende Informationen übergeben:

bei PARMOD=31: die Ein-/Ausgabe wird über die 31-Bit-Operandenliste angestoßen, die Informationen werden in den Registern 2, 3 und 4 übergeben:

Register	Inhalt
2	enthält den Ereignis-Informationscode
3	enthält in den beiden rechten Byte einen vom Benutzer beim Anstarten der Ein-/Ausgabe mitgelieferten POST-CODE und im linken Byte ein Kennzeichen für UPAM-Event (X'10')
4	enthält die Adresse der Operandenliste der beendeten Operation

Tabelle 72: Versorgung von Registern bei TU-Eventing (31-Bit-Operandenliste bei UPAM)

bei PARMOD=24: Die Ein-/Ausgabe wird über die „alte“ 24-Bit-Operandenliste angestoßen, die Informationen befinden sich in den Registern 2 und 3:

Register	Inhalt
2	enthält den Ereignis-Informationscode
3	enthält in den drei rechten Byte die Adresse der Operandenliste der beendeten Operationund im linken Byte den Wert X'10'

Tabelle 73: Versorgung von Registern bei TU-Eventing (24-Bit-Operandenliste bei UPAM)

Wird über eine mit PARMOD=31 erzeugte PAM-Operandenliste ein mit PARMOD=24 erzeugter SOLSIG-Makro oder eine 24-Bit-Contingency-Definition angesprochen, enthält der sekundäre Returncode (linkes Byte von Register 15) einen Hinweis auf inkonsistente Längen von Sender und Empfänger.

Aufbau des Steuerblocks FECB (File Event Control Block)

Der FECB muss auf Wortgrenze ausgerichtet werden. Er kann mit dem Makroaufruf IDECB mit symbolischen Namen versehen werden.

Feldbedeutung	Feldlänge (Byte)	Feldname
interne Kurzkennung der Ereigniskennung	4	CBEVID
Adresse des FCB	4	CBP1LNK
Standard-Gerätebyte	1	CBSDB
Fehlerbytes	3 x 1	CBSB1, CBSB2, CBSB3
Ablaufteil-Markierungsbyte	1	CBEFB
Anzahl übertragener PAM-Seiten	1	CBNPA

Tabelle 74: Aufbau des Eventing-Steuerblocks FECB (UPAM)

Ablaufteilmarkierungsbyte

Eine UPAM-Ein-/Ausgabe-Operation kann auf verschiedene Arten beendet werden (AMB: Ablaufteil-Markierungsbyte, siehe FECB):

normale Ein-/Ausgabe-Beendigung	AMB=X'80'
Ein-/Ausgabe-Operation führte zu Sonderbedingung	AMB=X'C0'
nicht behebbarer Fehler (z.B. Hardwarefehler)	AMB=X'A0'

In einem Contingency-Prozess kann der Benutzer auf die einzelnen Möglichkeiten der Ein-/Ausgabe-Beendigung entsprechend reagieren.

23.8 Multi-User-Betrieb

Eine UPAM-Datei kann mit folgenden Zugriffsmethoden erstellt bzw. bearbeitet werden:

- UPAM
- FASTPAM (siehe Kapitel „FASTPAM – Fast Primary Access Method“ (FASTPAM - Fast Primary Access Method))
- DIV (siehe Kapitel „DIV – Data In Virtual“ (DIV - Data In Virtual))

FASTPAM und DIV können jedoch nur UPAM-Dateien mit der Eigenschaft BLKCTRL=NO bearbeiten.

Die Erlaubnis für eine parallele Dateibearbeitung ist von den bei der Eröffnung angegebenen Operandenwerten für SHARUPD, MODE, LOCKENV und LOCVIEW abhängig.
Die möglichen parallelen Eröffnungen werden in der folgenden Tabelle dargestellt:

Möglichkeiten der parallelen Eröffnung

			USER B								
	SHARUPD=		***YES**			***NO**			***WEAK**		
		OPEN-Modus	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A	***YES**	INPUT INOUT OUTIN	X O O	O O O		X			X X X		
	***NO**	INPUT INOUT OUTIN	X			X			X X X		
	***WEAK**	INPUT INOUT OUTIN	X	X		X	X		X X X	X	

Tabelle 75: UPAM: erlaubte SHARUPD-/OPEN-Kombinationen

X: OPEN erlaubt

O: OPEN nur erlaubt,

- wenn die Eröffner dieselbe blockorientierte Zugriffsmethode benutzen (nur UPAM/FASTPAM oder nur DIV)
- **und** denselben Wert für den Operanden LOCKENV benutzen (alle LOCKENV=*HOST oder LOCKENV=*XCS)
- **und** alle im selben HOST laufen *oder* im selben XCS-Verbund bei Verwendung von LOCKENV=*XCS

Hinweise

- Leseoperationen mit SHARUPD=*WEAK können eine Datei gleichzeitig mit jeder beliebigen Schreiboperation eröffnet haben.

  Ausnahme:
  Leseoperationen mit DIV-SHARUPD=*WEAK, die bei OPEN LOCVIEW=*NONE spezifiziert haben, besitzen dieselbe Verträglichkeit wie Leseoperationen mit UPAM/FASTPAM-SHARUPD=*WEAK.
- Eröffner mit DIV-SHARUPD=*YES sind nicht mit Eröffnern mit UPAM/FASTPAM SHARUPD=*YES verträglich.
- Leseoperationen sind immer miteinander verträglich (unabhängig von Zugriffsmethode, SHARUPD-Spezifikation, LOCKENV-Spezifikation und Host).
- Nicht erlaubte Kombinationen führen zu einem OPEN-Fehler.
- Der Versuch, eine Band-Datei mit SHARUPD=YES oder WEAK zu eröffnen, führt ebenfalls zu einem OPEN-Fehler.
- Ist für eine PAM- oder SAM-Plattendatei ohne PAM-Schlüssel die Sekundärallokierung kleiner als BLKSIZE, so wird jeder Öffnungsversuch mit FCBTYPE=PAM ebenfalls von UPAM mit OPEN-Fehler abgewiesen. Ausnahmen: Sekundärallokierung=0 oder OPEN-Modus INPUT.
- Ist für eine Banddatei ohne PAM-Schlüssel BLKSIZE kein Vielfaches von 2048, so wird jeder Öffnungsversuch mit FCBTYPE=PAM mit OPEN-Fehler abgewiesen.
- Bei Zugriffen auf Dateien im Modus SHARUPD=YES kann der Fall eintreten, dass eine Datei mit einer Dateigröße < 32 GB durch entsprechende Verarbeitung zu einer Datei >= 32 GB wird.

  Hier werden zwei Fälle unterschieden:

  - Aufrufer, die auf diese Situation vorbereitet sind
    (mit der Angabe LARGE_FILE=*ALLOWED beim Makro FCB bzw.
    EXCEED-32GB=*ALLOWED beim Kommando ADD-FILE-LINK)
  - nicht vorbereitete Aufrufer
    (Angabe LARGE_FILE=*FORBIDDEN beim Makro FCB bzw.
    EXCEED-32GB=*FORBIDDEN beim Kommando ADD-FILE-LINK).

  Es wird nach jedem Aufruf des Allocators die Größe der betroffenen Datei überprüft. Wenn bei dieser Überprüfung eine Dateigröße >= 32 GB ermittelt wird und im zugehörigen FCB das Attribut LARGE_FILE=*FORBIDDEN gesetzt ist, wird die Verarbeitung abgebrochen. UPAM liefert in diesem Fall folgenden Returncode (bzw. die entsprechende DMS-Meldung `DMS09AD`):

  ```
  X'000009AD' FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT.
  ```
- Der Last Byte Pointer gibt das bytegenaue logische Dateiende einer PAM-Datei an. Er wird in den Katalogeintrag der Datei übernommen, wenn die Datei geschlossen wird. (Details siehe Abschnitt „PAM-Datei eröffnen (OPEN-Modi)“). Der Benutzer bzw. die Anwendung muss bei Shared-Update-Verarbeitung sicherstellen, dass der Auftrag, der zuletzt in den letzten logischen Block der Datei schreibt, die Datei auch zuletzt schließt. Andernfalls ist der LBP nicht korrekt versorgt, was beim Umkopieren sowie Sichern und Restaurieren zu Inkonsistenzen führen kann.

24 Anhang

Im Anhang sind folgenden Tabellen und Übersichten enthalten:

- Kennsatzformate (ab "Kennsatzformate")
  - Bandanfangs-Kennsätze
  - Benutzer-Bandanfangs-Kennsätze (UVL1 bis UVL9)
  - Dateianfangs-Kennsätze (HDR1 bis HDR9)
  - Benutzer-Dateianfangs-Kennsätze (UHL)
  - Bandende-Kennsätze (EOV1 bis EOV9)
  - Dateiende-Kennsätze (EOF1 bis EOF9)
  - Benutzer-Dateiende-Kennsätze (UTL)
- Verarbeitung der Kennsatzfelder (ab "Verarbeitung der Kennsatzfelder")
  - Anforderungen an ein sendendes System
  - Anforderungen an ein empfangendes System

24.1 Kennsatzformate

- Familie der Bandanfangs-Kennsätze
- Familie der Benutzer-Bandanfangs-Kennsätze (UVL1 bis UVL9)
- Familie der Dateianfangs-Kennsätze (HDR1 bis HDR9)
- Familie der Benutzer-Dateianfangs-Kennsätze (UHL)
- Familie der Bandende-Kennsätze (EOV1 bis EOV9)
- Familie der Dateiende-Kennsätze (EOF1 bis EOF9)
- Familie der Benutzer-Dateiende-Kennsätze (UTL)

24.1.1 Familie der Bandanfangs-Kennsätze

Jeder Datenträger enthält mindestens einen Bandanfangs-Kennsatz (VOL1) und höchstens neun. Die Bandanfangs-Kennsätze VOL2 bis VOL9 sind optional.

Erster Bandanfangs-Kennsatz (VOL1)

Der erste Bandanfangs-Kennsatz identifiziert den Datenträger, den Eigentümer, die Zugriffsbedingungen, die Implementation und die Ausgabenummer der verwendeten Norm.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	VOL
4	Kennsatznummer	1	1
5 bis 10	Bandkennzeichen	6	„a“-Zeichen. Vom Eigentümer fest zugeordnet, um das Band zu kennzeichnen
11	Datenträger-Zugriffsvermerk	1	„a“-Zeichen. Zeigt Einschränkungen bezüglich des Zugriffs zu den Daten auf diesem Band an. Ein Leerzeichen bedeutet, dass keine Einschränkung für den Zugriff auf den Datenträger gegeben ist. Ein beliebiges anderes „a“-Zeichen bedeutet, dass es besondere Einschränkungen für den Zugriff auf den Datenträger gibt. BS2000: • Leerzeichen oder '0' : unbeschränkter Zugriff • '1' : Zugriff nur vom Eigentümer möglich
12 bis 24	Reserviert für spätere Normung	13	Leerzeichen
25 bis 37	System-Code	13	„a“-Zeichen. Identifiziert die Implementation, die die Bandanfangs-Kennsätze aufgezeichnet hat.
38 bis 51	Eigentümer-Kennzeichen	14	„a“-Zeichen. Identifiziert den Eigentümer des Bandes. Für BS2000:
38 bis 41		4	Leerzeichen
42 bis 49		8	„a“-Zeichen: Benutzerkennung
50 bis 51		2	Leerzeichen
52 bis 79	Reserviert für spätere Normung	28	Leerzeichen

80	Normvermerk	1	Zeigt die Ausgabe der Norm an, der die Kennsätze und Datenformate auf diesem Band entsprechen. • 4 bedeutet: DIN 66029-4 (Ausgabe September 1987) • 3 bedeutet: DIN 66029-3 (Ausgabe Mai 1979) • 2 bedeutet: DIN 66029-2 (Ausgabe Juni 1976) • 1 bedeutet: DIN 66029-1 (Ausgabe August 1972)

Weitere Bandanfangs-Kennsätze (VOL2 bis VOL9)

Die weiteren Bandanfangs-Kennsätze sind optional. Die Implementationen von Fujitsu erzeugen sie nicht.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	VOL
4	Kennsatznummer	1	Ziffer 2 bis 9
5 bis 80	Reserviert für die Implementation	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

24.1.2 Familie der Benutzer-Bandanfangs-Kennsätze (UVL1 bis UVL9)

Die Benutzer-Bandanfangs-Kennsätze sind optional.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	UVL
4	Kennsatznummer	1	Ziffer 2 bis 9
5 bis 80	Reserviert für den Betreiber	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

BS2000 liefert die UVL-Kennsätze an den Benutzer.

24.1.3 Familie der Dateianfangs-Kennsätze (HDR1 bis HDR9)

Jede Datei oder jeder Dateiabschnitt enthält mindestens zwei Dateianfangs-Kennsätze (HDR1 und HDR2) und höchstens neun. Die Dateianfangs-Kennsätze HDR3 bis HDR9 sind optional.

Erster Dateianfangs-Kennsatz (HDR1)

Der erste Dateianfangs-Kennsatz identifiziert einen Dateiabschnitt, beschreibt seine Lage innerhalb der Dateimenge und bestimmt gewisse Merkmale des Dateiabschnitts.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	HDR
4	Kennsatznummer	1	1
5 bis 21	Dateiname	17	„a“-Zeichen. Kennzeichnet die Datei.
22 bis 27	Dateimengen-Kennzeichen	6	„a“-Zeichen. Kennzeichnet die Dateimenge, zu der diese Datei gehört.
28 bis 31	Dateiabschnittsnummer	4	„n“-Zeichen. Kennzeichnet den Dateiabschnitt. Die Nummer des ersten Dateiabschnitts einer Datei ist 0001. Diese Nummer wird für jeden folgenden Dateiabschnitt dieser Datei um eins erhöht.
32 bis 35	Dateifolgenummer	4	„n“-Zeichen. Kennzeichnet die Datei der Dateimenge. Die Dateifolgenummer der ersten Datei in einer Dateimenge ist 0001. Diese Nummer wird bei jeder folgenden Datei einer Dateimenge um eins erhöht. In allen Kennsätzen einer bestimmten Datei muss dieses Feld die gleiche Zahl enthalten, unabhängig davon, ob die Datei auf einem oder mehreren Bändern liegt.
36 bis 39	Generationsnummer	4	„n“-Zeichen. Unterscheidet die aufeinander folgenden Fortschreibungen der Datei von 0001 bis 9999.
40 bis 41	Versionsnummer	2	„n“-Zeichen. Unterscheidet die aufeinander folgenden Wiederholungen einer Generation
42 bis 47	Erstellungsdatum	6	Leerzeichen oder „n“-Zeichen. Gibt das Datum der Erstellung des Dateiabschnitts an. Ein Leerzeichen für das 20. Jahrhundert; die Ziffer 0 für das 21. Jahrhundert, gefolgt von zwei „n“-Zeichen für das Jahr (00 bis 99) innerhalb des Jahrhunderts, gefolgt von drei „n“-Zeichen für den Tag des Jahres (001 bis 366). Der Wert 00000 in den letzten fünf Stellen zeigt an, dass das Erstellungsdatum ohne Bedeutung ist.

48 bis 53	Verfallsdatum	6	Leerzeichen oder „n“-Zeichen. Gibt das früheste Datum an, ab dem der Dateiabschnitt gelöscht werden darf. Format wie Feld Erstellungsdatum (Stelle 42 bis 47). Der Wert 00000 in den letzten fünf Stellen zeigt an, dass das Verfallsdatum ohne Bedeutung ist und der Dateiabschnitt veraltet ist.
54	Dateizugriffsvermerk	1	„a“-Zeichen. Zeigt Einschränkungen bezüglich des Zugriffs zu den Daten dieser Datei an. Ein Leerzeichen bedeutet, dass keine Einschränkung für den Zugriff auf die Datei gegeben ist. Ein beliebiges anderes „a“-Zeichen bedeutet, dass es besondere Einschränkungen für den Zugriff auf die Datei gibt. BS2000: bei '1' oder '3' haben Band- oder Dateieigentümer Zugriff
55 bis 60	Blockzähler	6	000000
61 bis 73	System-Code	13	„a“-Zeichen. Identifiziert die Implementation, durch die die Kennsätze erzeugt werden.
61 bis 65		5	BS2000: Leerzeichen
66 bis 73		8	BS2000: Leerzeichen
74 bis 80	Reserviert für spätere Normung	7	Leerzeichen

Zweiter Dateianfangs-Kennsatz (HDR2)

Der zweite Dateianfangs-Kennsatz beschreibt Merkmale der Datei und der Implementation.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	HDR
4	Kennsatznummer	1	2
5	Satzformat	1	F, D oder S und lt. DIN nicht unterstützt V und U. Gibt das Format der Sätze der Datei an. • F: alle Sätze der Datei haben eine feste Satzlänge • D: alle Sätze haben eine variable Länge und die Anzahl der Zeichen als Dezimalzahl ist im Satz selbst angegeben • S: alle Sätze sind segmentiert • U: alle Sätze haben eine undefinierte Länge (lt. DIN nicht unterstützt) • V: alle Sätze haben eine variable Länge und die Anzahl der Zeichen als Dualzahl ist im Satz selbst angegeben (lt. DIN nicht unterstützt)
6 bis 10	Blocklänge	5	„n“-Zeichen. Gibt die maximale Anzahl der Zeichen je Block der Datei an. BS2000: Bei Normvermerk 1 kann der Inhalt wie folgt sein:
6 bis 7	STD-Blöcke		'80': Kennzeichen für STD-Block
8 bis 10			„n“-Zeichen. Gibt die Anzahl der STD-Blöcke an.
11 bis 15	Satzlänge	5	„n“-Zeichen. Kennzeichnet die Satzlänge in Verbindung mit dem Satzformat (Stelle 5): • Bei Satzformat F enthält dieses Feld die tatsächliche Satzlänge. • Bei Satzformat D und V enthält dieses Feld die maximale Länge einschließlich Satzlängenwort (SLW). • Bei Satzformat S enthält dieses Feld die maximale Satzlänge, wobei die Segmentkontrollwörter (SKW) ausgenommen sind. Der Inhalt 00000 in diesem Feld bedeutet beim Satzformat S, dass die Satzlänge größer als 99999 sein kann. • Bei Satzformat U enthält dieses Feld die maximale Anzahl der Zeichen, die ein Satz enthalten kann.
16 bis 50	Reserviert für die Implementation	35	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldesgibt es weder Festlegungen noch Einschränkungen.

16	Schreibdichte	1	Belegung von BS2000 bis zur Unterstützung der DIN Stufe 4: 0 = 200 Bpi 1 = 556 Bpi 2 = 800 Bpi 3 = 1600 Bpi 4 = 6250 Bpi
17	Datenposition	1	Anzeige bei Spulenwechsel 0 = nein 1 = ja
18 bis 34	Auftragsschrift-Kennung	17	Durch Prozessverwalter zugewiesene Kennung
35 bis 36	Schreibdichte bei MBKs	2	2 Leerzeichen = nicht komprimiert 'P' und 1 Leerzeichen = komprimiert
47 bis 50	Dateinamencode	4	Wird nur bis DIN 66029-1 verwendet, wenn die Stellen 6 bis 7 STD-Blöcke enthalten.
51 bis 52	Pufferverschiebung	2	„n“-Zeichen. Gibt die Länge (in Zeichen) eines zusätzlichen Feldes an, das am Anfang eines jeden Datenblocks eingefügt ist.
53 bis 80	Reserviert für spätere Normung	28	Leerzeichen

Dritter Dateianfangs-Kennsatz (HDR3)

Der HDR3-Kennsatz enthält für den Dateieigentümer den vollständigen Dateinamen, die Kennwörter und die Zugriffsart.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	HDR
4	Kennsatznummer	1	3
5 bis 12	Eigentümerkennzeichen	8	Identifiziert den Eigentümer der Datei (Benutzerkennung).
13 bis 56	Dateiname	44	Die ersten 44 Zeichen des Namens der Datei bzw. der Dateigeneration, zu der die Datei gehört.
57 bis 60	Lesekennwort	4	Bezeichnet ein zum Lesen der Datei erforderliches Kennwort
61 bis 64	Schreibkennwort	4	Bezeichnet ein zum Lesen und Schreiben der Datei erforderliches Kennwort
65 bis 68	Ablaufkennwort	4	Bezeichnet ein, um einen in der Datei befindlichen Lademodul ablaufen zu lassen, erforderliches Kennwort
69	Zugriffsart	1	Gibt die zulässige Zugriffsart an: • 0: Lese- und Schreibzugriff erlaubt • 1: nur Lesezugriff erlaubt
70 bis 80	Reserviert	11	Leerzeichen

Weitere Dateianfangs-Kennsätze (HDR4 bis HDR9)

Die weiteren Dateianfangs-Kennsätze enthalten implementationsabhängige Informationen.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	VOL
4	Kennsatznummer	1	Ziffer 2 bis 9
5 bis 80	Reserviert für die Implementation	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

24.1.4 Familie der Benutzer-Dateianfangs-Kennsätze (UHL)

Die Benutzer-Dateianfangs-Kennsätze sind optional.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	UHL
4	Kennsatznummer	1	„a“-Zeichen. Vom Benutzer festzulegen. BS2000: „b“-Zeichen
5 bis 80	Reserviert für den Benutzer	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

BS2000 unterstützt bis zu 255 UHL-Kennsätze. Die Kennsatznummer (Stelle 4) wird vom Benutzer festgelegt.

24.1.5 Familie der Bandende-Kennsätze (EOV1 bis EOV9)

Jeder Dateiabschnitt, der auf einer Folgespule fortgesetzt wird, enthält mindestens zwei Bandende-Kennsätze (EOV1 und EOV2) und höchstens neun. Die Bandende-Kennsätze EOV3 bis EOV9 sind optional.

Erster Bandende-Kennsatz (EOV1)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOV
4	Kennsatznummer	1	1
5 bis 54	gleich den entsprechenden Feldern in HDR1	50	gleich den entsprechenden Feldern in HDR1
55 bis 60	Blockzähler	6	„n“-Zeichen. Gibt die Anzahl der Datenblöcke an, die den Dateiabschnitt bilden
61 bis 80	gleich den entsprechenden Feldern in HDR1	20	gleich den entsprechenden Feldern in HDR1

Zweiter Bandende-Kennsatz (EOV2)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOV
4	Kennsatznummer	1	2
5 bis 80	gleich den entsprechenden Feldern in HDR2	76	gleich den entsprechenden Feldern in HDR2

Dritter Bandende-Kennsatz (EOV3)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOV
4	Kennsatznummer	1	3
5 bis 80	gleich den entsprechenden Feldern HDR3	76	gleich den entsprechenden Feldern in HDR3

Weitere Bandende-Kennsätze (EOV4 bis EOV9)

Die Bandende-Kennsätze enthalten implementationsabhängige Informationen.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	UVL
4	Kennsatznummer	1	4 bis 9
5 bis 80	Reserviert für den Betreiber	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

24.1.6 Familie der Dateiende-Kennsätze (EOF1 bis EOF9)

Jede Datei enthält mindestens zwei Dateiende-Kennsätze (EOF1 und EOF2) und höchstens neun. Die Dateiende-Kennsätze EOF3 bis EOF9 sind optional.

Erster Dateiende-Kennsatz (EOF1)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOF
4	Kennsatznummer	1	1
5 bis 54	gleich den entsprechenden Feldern in HDR1	50	gleich den entsprechenden Feldern in HDR1
55 bis 60	Blockzähler	6	„n“-Zeichen. Gibt die Anzahl der Datenblöcke an, die den Dateiabschnitt bilden
61 bis 80	gleich den entsprechenden Feldern in HDR1	20	gleich den entsprechenden Feldern in HDR1

Zweiter Dateiende-Kennsatz (EOF2)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOF
4	Kennsatznummer	1	2
5 bis 80	gleich den entsprechenden Feldern in HDR2	76	gleich den entsprechenden Feldern in HDR2

Dritter Dateiende-Kennsatz (EOF3)

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOF
4	Kennsatznummer	1	2
5 bis 80	gleich den entsprechenden Feldern in HDR2	76	gleich den entsprechenden Feldern in HDR2

Weitere Dateiende-Kennsätze (EOF4 bis EOF9)

Die weiteren Dateiende-Kennsätze enthalten implementationsabhängige Informationen.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	EOF
4	Kennsatznummer	1	4 bis 9
5 bis 80	Reserviert für die Implementation	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

24.1.7 Familie der Benutzer-Dateiende-Kennsätze (UTL)

Die Benutzer-Dateiende-Kennsätze sind optional.

Aufbau

Stelle	Feldname	Länge	Feldinhalt
1 bis 3	Kennsatzname	3	UTL
4	Kennsatznummer	1	„a“-Zeichen. Vom Benutzer festzulegen BS2000: „b“-Zeichen
5 bis 80	Reserviert für den Betreiber	76	Für Aufzeichnung und inhaltliche Bedeutung dieses Feldes gibt es weder Festlegungen noch Einschränkungen

BS2000 unterstützt bis 255 UTL-Kennsätze. Die Kennsatznummern (Stelle 4) werden vom Benutzer festgelegt.

24.2 Verarbeitung der Kennsatzfelder

- Anforderungen an ein sendendes System
- Anforderungen an ein empfangendes System

24.2.1 Anforderungen an ein sendendes System

Dateien

Das Anwendungsprogramm muss die Datensätze der Dateien an die Implementation übergeben.

Kennsätze

Der Benutzer muss die in jedem der nachfolgend aufgeführten Kennsatzfelder benötigte Information zur Aufzeichnung an die Implementation übergeben, andernfalls muss die Implementation diese Information liefern.

Für jeden Datenträger einer Datenträgermenge:

- Bandkennzeichen (VOL1, Stelle 5 bis 10)
- Datenträgerzugriffsvermerk (VOL1, Stelle 11)

Für jede Datei einer Dateimenge:

- Dateizugriffsvermerk (HDR1, Stelle 54)

Wenn die Implementation dem Benutzer erlaubt, die Information bereitzustellen, die in jedem der nachfolgend aufgeführten Kennsatzfelder zu verarbeiten ist, dann muss die Implementation diese Information verarbeiten. Wenn der Benutzer diese Information nicht liefert, muss die Implementation die Information bereitstellen.

Für jeden Datenträger einer Datenträgermenge:

- Eigentümerkennzeichen (VOL1, Stelle 38 bis 51)

Für jede Datei einer Dateimenge:

- Dateimengenkennzeichen (HDR1, Stelle 22 bis 27)

Die Implementation muss dem Anwendungsprogramm erlauben, die Information bereitzustellen, die in jedem der nachfolgend aufgeführten Kennsatzfelder zu verarbeiten ist. Wenn das Anwendungsprogramm diese Information nicht liefert, muss die Implementation die Information für das jeweilige Feld bereitstellen.

Für jede Datei einer Dateimenge:

- Dateiname (HDR1, Stelle 5 bis 21)
- Satzformat (HDR2, Stelle 5)
- Blocklänge (HDR2, Stelle 6 bis 10)
- Satzlänge (HDR2, Stelle 11 bis 15)

Wenn die Implementation dem Anwendungsprogramm erlaubt, die Information bereitzustellen, die in jedem der nachfolgend aufgeführten Kennsatzfelder zu verarbeiten ist, dann muss die Implementation diese Information verarbeiten. Wenn das Anwendungsprogramm diese Information nicht liefert, muss die Implementation die Information bereitstellen.

Für jede Datei einer Dateimenge:

- Generationsnummer (HDR1, Stelle 36 bis 39)
- Versionsnummer (HDR1, Stelle 40 und 41)

Für jeden Dateiabschnitt einer Dateimenge:

- Erstellungsdatum (HDR1, Stelle 42 bis 47)

- Verfallsdatum (HDR1, Stelle 48 bis 53)

Wenn die Implementation in der Lage ist, eine Familie von Benutzer-Bandanfangs-Kennsätzen (UVL) zu verarbeiten, dann muss die Implementation dem Benutzer erlauben, die zu verarbeitende Information aus den nachfolgend aufgeführten Kennsatzfeldern bereitzustellen. Die Verarbeitung der entsprechenden Kennsätze wird nicht gefordert, wenn der Benutzer die Information nicht bereitstellt.

Für jeden Kennsatz aus einer Familie von Benutzer-Bandanfangs-Kennsätze, die auf jedem Band einer Bandmenge aufgezeichnet sind, gilt:

- Reserviert für den Benutzer (UVL, Stelle 5 bis 80)

Wenn die Implementation in der Lage ist, eine Familie von Benutzer-Dateianfangs-Kennsätzen (UHL) oder von Benutzer-Dateiende-Kennsätzen (UTL) zu verarbeiten, dann muss die Implementation dem Anwendungsprogramm erlauben, die Information bereitzustellen, die in den nachfolgend aufgeführten Kennsatzfeldern für eine Kennsatzfamilie einzutragen ist. Die Verarbeitung der entsprechenden Kennsätze wird nicht gefordert, wenn das Anwendungsprogramm die Informationen nicht bereitstellt.

Für jeden Kennsatz in einer Familie von Benutzer-Dateianfangs- und Benutzer-Dateiende-Kennsätze eines Magnetbandes gilt:

- Kennsatznummer (UHL/UTL, Stelle 4)
- Reserviert für den Benutzer (UHL/UTL, Stelle 5 bis 80)

Die Implementation kann dem Benutzer bezüglich der Satzlänge (HDR2, Stelle 11 bis 15) die nachfolgend beschriebenen Beschränkungen auferlegen.

Wenn die Implementation segmentierte Sätze verarbeitet, kann sie eine maximale Satzlänge festlegen. Diese Grenze soll nicht unter der maximalen zulässigen Blocklänge liegen, abzüglich der Länge des Pufferverschiebungsfeldes und abzüglich der Länge des Segmentkontrollwortes (SKW).

Wenn die Implementation Sätze variabler Länge verarbeitet, kann sie eine maximale Satzlänge festlegen, die der maximalen Blocklänge, abzüglich der Länge des Pufferverschiebungsfeldes und abzüglich der Länge des Satzlängenwortes (SLW) entspricht.

24.2.2 Anforderungen an ein empfangendes System

Dateien

Die Implementation muss für das Anwendungsprogramm den Inhalt der Datensätze und die Länge jedes Datensatzes bereitstellen. Das Segmentkontrollwort (SKW) und das Satzlängenwort (SLW) sind nicht Bestandteile des Datensatzes.

Kennsätze

Die Implementation muss dem Benutzer gestatten, ausreichende Information bereitzustellen, um die von ihm angeforderten Dateien sowie den Datenträger auszuwählen, auf dem diese Dateien aufgezeichnet sind.

Die Implementation muss für den Betreiber die Information bereitstellen, die in den nachfolgend aufgeführten Kennsatzfeldern enthalten ist.

Für jeden Datenträger einer Datenträgermenge:

- Bandkennzeichen (VOL1, Stelle 5 bis 10)
- Datenträgerzugriffsvermerk (VOL1, Stelle 11)

Für jede Datei einer Dateimenge:

- Dateizugriffsvermerk (HDR1, Stelle 54)

Die Implementation muss für das Anwendungsprogramm die Information bereitstellen, die in den nachfolgend aufgeführten Kennsatzfeldern enthalten ist.

Für jede Datei einer Dateimenge:

- Dateiname (HDR1, Stelle 5 bis 21)
- Satzformat (HDR2, Stelle 5)
- Blocklänge (HDR2, Stelle 6 bis 10)
- Satzlänge (HDR2, Stelle 11 bis 15)

Die Implementation braucht für den Benutzer nicht die Information bereitzustellen, die in den nachfolgenden Kennsatzfeldern enthalten ist.

Für jeden Datenträger einer Datenträgermenge:

- Eigentümerkennzeichen (VOL1, Stelle 38 bis 51)

Für jede Datei einer Dateimenge:

- Dateimengenkennzeichen (HDR1, Stelle 22 bis 27)
- Generationsnummer (HDR1, Stelle 36 bis 39)
- Versionsnummer (HDR1, Stelle 40 und 41)

Für jeden Dateiabschnitt einer Dateimenge:

- Erstellungsdatum (HDR1, Stelle 42 bis 47)
- Verfallsdatum (HDR1, Stelle 48 bis 53)

Wenn die Implementation in der Lage ist, dem Benutzer die Information zur Verfügung zu stellen, die in der Familie der Benutzer-Bandanfangs-Kennsätze (UVL) aufgezeichnet ist, dann muss die Information bereitgestellt werden, die in den nachfolgenden Kennsatzfeldern enthalten ist.

Für jeden Kennsatz einer Familie von Benutzer-Bandanfangs-Kennsätzen:

- Reserviert für den Benutzer (UVL, Stelle 5 bis 80)

Wenn die Implementation in der Lage ist, dem Benutzer die Information zur Verfügung zu stellen, die in der Familie von Benutzer-Dateianfangs-Kennsätzen (UHL) oder von Benut-zer-Dateiende-Kennsätzen (UTL) aufgezeichnet ist, dann muss die Information bereitgestellt werden, die in den nachfolgenden Kennsatzfeldern enthalten ist.

- Kennsatznummer (UHL/UTL, Stelle 4)
- Reserviert für den Benutzer (UHL/UTL, Stelle 5 bis 80)

25 Fachwörter

Im Folgenden werden kurze Definitionen einiger in der vorliegenden Beschreibung verwendeter Fachbegriffe gegeben.

Abschnittsmarke (Tape Mark)

Ein Bandblock, der die Grenze sowohl zwischen Datenblöcken und Kennsatzgruppen als auch zwischen bestimmten Kennsatzgruppen anzeigt. Der Aufbau der Abschnittsmarke ist in den entsprechenden Normen über Magnetbänder angegeben.

ACL (Access Control List)

Der Dateischutz mit einer ACL (Zugriffskontroll-Liste) wird bereits seit SECOS V4.0 nicht mehr unterstützt. Das Dateimerkmal ACL ist noch im Katalogeintrag enthalten, enthält aber im Normalfall den Wert NO (kein ACL-Schutz).

Sollte der Fall auftreten, dass eine Datei ACL-geschützt ist (evtl. möglich in einem sehr alten Datenbestand), ist der Dateizugriff nicht möglich. In diesem Fall kann der Eigentümer die Datei mit GUARDS schützen. Der Schutz durch GUARDS „überschreibt“ den ACL-Schutz und macht die Datei damit wieder zugänglich.

Alias Catalog Service (ACS)

Dateien und Jobvariablen können über Aliasnamen angesprochen werden und zusammen mit der Zuordnung zur realen Datei/JV in speziellen Katalogen, den Aliaskatalogen, hinterlegt werden. Der Alias Catalog Service (ACS) umfasst drei Grundfunktionen: Aliasnamen-Vereinbarung, Catid-Einfügung für temporäre Spooldateien und Präfix-Einfügung.

alphanumerisch

alphanumerische Zeichen umfassen *alphabetische* und *numerische* Zeichen, d.h. die Buchstaben A-Z und die Ziffern 0-9.

Archivnummer (= VSN, Volume Serial Number)

Sie besteht aus 6 Zeichen und wird dem Datenträger bei der Initialisierung (VOLIN bzw. INIT) zugeteilt. Sie ist im Standard-Datenträgerkennsatz enthalten und dient der Identifizierung des Datenträgers.

Auftragsnummer (= TSN, Task Sequence Number)

die vom System für den Auftrag vergebene (laufende) Nummer, mit der der Anwender bei einigen Kommandos einen Auftrag identifizieren kann.

Auftrag

Gesamtheit aller Abläufe zwischen den Benutzerkommandos SET-LOGON-PARAMETERS bzw. LOGON und EXIT-JOB bzw. LOGOFF. Dabei ist es unwichtig, ob der Auftrag bereits vollständig definiert ist (Batchbetrieb) oder ob die einzelnen Schritte erst beim Ablauf festgelegt werden (Dialogbetrieb).

Band (Volume)

Eine auswechselbare Einheit des Datenträgers Magnetband. Ein Band kann eine Datei ganz oder teilweise enthalten. Es kann auch mehrere Dateien und/oder einen bzw. mehrere Datei-Abschnitte enthalten.

Bandmenge (Volume Set)

Das Band oder die Bänder, auf denen die Dateien einer Dateimenge aufgezeichnet sind.

Batchbetrieb

Auftrag, der mit dem ENTER-JOB- bzw. ENTER-PROCEDURE-Kommando gestartet wurde; im Gegensatz zum Dialogbetrieb ist der Ablauf vordefiniert und in einer ENTER-Datei (Start mit ENTER-JOB) bzw. in einer Prozedurdatei (Start mit ENTER-PROCEDURE) festgelegt.

Batchverarbeitung

Siehe Batchbetrieb

Block

Siehe PAM-Seite, Datenblock

BS2000-Datei

Bezeichnet eine Datei, die ausschließlich von BS2000 angelegt und bearbeitet wird. BS2000-Dateien auf Net-Storage (FILE-TYPE=BS2000) werden seit BS2000/OSD-BC V9.0 bedient. Sie liegen direkt auf einem Net-Storage-Volume. Offene Systeme dürfen ausschließlich lesend darauf zugreifen.

CALL-Prozedur

Kommando-/Anweisungsfolge, die in einem Auftrag abläuft; Aufruf mit CALL-PROCEDURE-Kommando (siehe Handbuch „Kommandos“ [3]).

Catid (= Katalogkennung)

Kennzeichen eines Pubsets (Siehe Pubset-Id); wird im vollständigen Dateinamen/Pfadnamen in der Form :catid: angegeben.

Datei

Sätze, die zueinander in Beziehung stehen, werden in einer benannten Einheit, der Datei, zusammengefasst. Dateien sind beispielsweise: konventionelle Ein/Ausgabedaten von Programmen; Lademodule und Bindemodulbibliotheken; Textinformation, die mit dem Dateiaufbereiter erstellt und verarbeitet wird.

Dateiabschnitt (File Section)

Der Teil einer Datei, der auf einem einzigen Band aufgezeichnet ist.

Dateikettungsname

maximal 8 Zeichen langer Name, der die Verbindung zwischen dem FCB-Makroaufruf oder Dateisteuerblock und der Datei über Task File Table herstellt.

Dateimenge (File Set)

Eine Menge von Dateien, die aufeinander folgend auf einem oder mehreren Bändern aufgezeichnet ist. Zwischen den Abschnitten einer Datei innerhalb einer Dateimenge dürfen sich keine Abschnitte anderer Dateien befinden.

Datenblock

Ein Block, der einen oder mehrere Sätze einer Datei enthält.

Dialogbetrieb

Ein Auftrag, der von einem entfernten Datenplatz eingeleitet wird und abläuft; der Ablauf ist nicht im Voraus festgelegt.

Doppel-Abschnittsmarke (Double Tape Mark)

Zwei unmittelbar aufeinander folgende Abschnittsmarken. Sie zeigen das logische Bandende an. Zwei aufeinander folgende Abschnittsmarken treten auch dann auf, wenn ein leerer Dateiabschnitt, oder eine leere Datei auf dem Band steht.
In diesem Fall werden sie nicht als Doppel-Abschnittsmarke interpretiert, sondern als zwei einfache Abschnittsmarken.

„Leer“ bedeutet: es gibt keine Datenblöcke zwischen den Abschnittsmarken, vor der Datei-Anfangskennsatzgruppe und nach der Datei-/Bandendekennsatzgruppe.

FCB (File Control Block)

Dateisteuerblock, der die für die Dateiverarbeitung benötigten Informationen enthält.

First in – first out (FIFO)

Warteschlangen-Struktur, die Informationen in der Reihenfolge der Eingabe abarbeitet (im Gegensatz dazu: Last in – first out, LIFO).

FOREIGN-Datei

Eine FOREIGN-Datei ist eine Datei auf einem privaten Datenträger oder auf einem Net-Storage-Volume, die aber nicht auf einem Pubset katalogisiert ist.

geblockter Satz (blocked record)

Satz in einer Datei, in der jeder Datenblock mehrere Sätze oder Satzsegmente enthalten kann.

Kennsatz (Label)

Ein Satz am Beginn oder Ende eines Bandes oder einer Datei, der dazu dient, Band oder Datei zu identifizieren, zu beschreiben und/oder zu begrenzen. Ein Kennsatz wird nicht als Bestandteil der Datei betrachtet. Jeder Kennsatz wird für sich in einem eigenen Block aufgezeichnet (Kennsatzblock).

Kennsatzfamilie (Label Set)

Eine ununterbrochene Folge von Kennsätzen mit demselben Kennsatznamen.

Kennsatzgruppe (Label Group)

Eine ununterbrochene Folge von Kennsatzfamilien, die ein Band, einen Dateiabschnitt oder eine Datei begrenzt.

Kennsatzname (Label Identifier)

Ein Wort aus drei Zeichen, das als Teil des Kennsatzes aufgezeichnet ist und ihn kennzeichnet.

Kennsatzroutine (Label Handling Routine)

Folge von Anweisungen zur Verarbeitung von Kennsätzen.

Klasse-1-Speicher

Der Teil des virtuellen Speichers, der von den hauptspeicherresidenten Moduln des Ablaufteils belegt ist. Alle Seiten der Klasse 1 sind als privilegiert und nichtseitenwechselbar gekennzeichnet. Von diesen Seiten ist auf dem Seitenwech-sel-Speicher kein Abbild enthalten. Die Seiten sind während des gesamten Systemlaufs im Hauptspeicher.

Klasse-5-Speicher

Der Teil des virtuellen Benutzerspeichers, der die für einen Benutzerauftrag erforderlichen seitenwechselbaren Bereiche enthält, die vom Ablaufteil dynamisch zugewiesen werden.

Klasse-6-Speicher

Der Teil des virtuellen Benutzerspeichers, der die Benutzerprogramme enthält, die vom Ablaufteil dynamisch zugewiesen werden.

Last Byte Pointer (LBP)

Zeiger auf das letzte gültige Byte des letzten logischen Blocks einer PAM-Datei.

Last Page Pointer (LPP)

Zeiger auf die letzte von einer Datei belegte PAM-Seite. Entspricht im Katalogeintragder Highest-Used-Page.

LBN (= Logical Block Number)

(laufende) Nummerierung der PAM-Seiten einer Datei

Locate-Mode

Im Locate-Mode fordert der Benutzer die Adresse des aktuellen Satzes an, der sich in einem Pufferbereich befindet. Für die Datenübertragung zum und vom Puffer ist der Benutzer zuständig.

mehrbenutzbare Datei

Eine Datei, die mit USER-ACCESS=ALL-USERS katalogisiert ist. Auf eine mehrbenutzbare Datei kann man von allen Benutzerkennungen aus zugreifen, sofern die übrigen Schutzattribute (z.B. Dateikennwörter) der Datei dies zulassen.

Move-Mode

Im Move-Mode gibt der Anwender die Lage des Satzes in seinem Programm an. Für die Datenübertragung zum und vom Puffer ist das System zuständig.

Net-Client

Realisiert den Zugriff auf Net-Storage für das nutzende Betriebssystem.
In BS2000 transformiert der Net-Client zusammen mit dem BS2000-Subsystem ONETSTOR die BS2000-Dateizugriffe in entsprechende UNIX-Dateizugriffe und führt sie über NFS auf dem Net-Server aus.
Net-Client bei SU /390 und S-Server ist der HNC, bei SU x86 und SQ-Server das Trägersystem X2000.

Net-Server

File-Server im weltweiten Rechnernetz, der Speicherplatz (Network Attached Storage, NAS) für die Nutzung durch andere Server bereitstellt und entsprechende File-Server-Dienste anbietet.

Net-Storage

Der von einem Net-Server im Rechnernetz bereitgestellte und zur Nutzung durch fremde Server freigegebene Speicherplatz. Net-Storage kann ein Dateisystem oder auch nur ein Knoten im Dateisystem des Net-Servers sein.

Net-Storage-Datei

Bezeichnet eine Datei, die auf einem Net-Storage-Volume angelegt ist. Auf Net-Storage wird zwischen den zwei Dateitypen BS2000-Datei und Node-File unterschieden.

Net-Storage-Volume

Net-Storage-Volumes repräsentieren Net-Storage im BS2000, die die Systembetreuung als Erweiterung von Daten-Pubsets bereitstellt.
Net-Storage-Volumes werden durch ihre Volume Serial Number (VSN) und den Volumetyp NETSTOR angesprochen. Die VSN des Net-Storage-Volumes entspricht im freigegebenen Dateisystem des Net-Servers dem Verzeichnis, das die Benutzerdateien und Metadaten enthält.

NFS (Network File System)

BS2000-Softwareprodukt, mit dem verteilte Datenhaltung in einem heterogenen Rechnernetz möglich ist. Der Benutzer kann auf ferne Dateien so zugreifen, als ob sie an seinem lokalen Rechner vorhanden wären.
NFS dient somit der Konnektivität zwischen Systemen. Außerdem können Dateien mit NFS automatisch und zuverlässig durch das BS2000 gesichert werden.

Node-File

Bezeichnet eine Net-Storage-Datei (FILE-TYPE=NODE-FILE), die sowohl von BS2000 als auch von offenen Systemen angelegt und bearbeitet werden kann. Node-Files werden ab BS2000 OSD/BC V10.0 unterstützt. Sie liegen auf einem Net-Storage-Volume in einem benutzerspezifischen Verzeichnis (Name der Benutzerkennung) und die Dateinamen entsprechen den BS2000-Namenskonventionen.

Nulldatei

Eine Datei, die logisch leer ist. Es handelt sich hier um eine Datei, die katalogisiert und der vom System Speicherplatz zugewiesen wurde, die jedoch keine Daten enthält.

Ortungsbetrieb

Im Ortungsbetrieb fordert der Benutzer die Adresse des aktuellen Satzes an, der sich in einem Pufferbereich befindet. Für die Datenübertragung zum und vom Puffer ist der Anwender zuständig.

PAM-Seite

Zusammenhängender Speicherplatz von 2048 Byte, beginnend an einer durch 2048 dividierbaren Adresse.

Privilegierter Modus/Programm

alle Teile des Betriebssystems, die nicht im unprivilegierten Verarbeitungszustand ablaufen.

Prozedur/Prozedurdatei

Datei, die eine festgelegte Kommando- oder Anweisungsfolge enthält, die der Programmeingabe dienen.
Prozeduren werden mit CALL-PROCEDURE bzw. ENTER-PROCEDURE gestartet. Nur wenn die Datei einen ENTER-Job enthält, wird sie mit ENTER-JOB gestartet. Näheres zu Prozeduren siehe Handbuch „Kommandos“ [3]).

Pubset

Satz gemeinschaftlich gekennzeichneter Platten. MPVS-Systeme arbeiten mit mehreren voneinander unabhängigen Pubsets.

Pubset-Id

Kennzeichen eines Pubsets. Beginnt die Archivnummer eines gemeinschaftlichen Datenträgers mit den drei Zeichen „PUB“, ist die Pubset-Kennung das 4. Zeichen, enthält sie einen Punkt, bilden die Zeichen vor dem Punkt die Pubset-Kennung. Im Pfadnamen wird die Pubset-Kennung in der Form : catid: (siehe Catid (= Katalogkennung)) angegeben.

Puffer

Zusammenhängender Speicherbereich: ein Teil des Hauptspeichers, aus dem Daten gelesen werden oder in den Daten geschrieben werden;

Public Volume Set (PVS)

veraltet für: Pubset; siehe Pubset

PVS-id

Siehe Pubset-Id

Satz

Eine Zusammenfassung von Daten, die als eine logische Einheit behandelt werden.

Satz fester Länge:
Ein Satz in einer Datei, in der alle Sätze nach Vereinbarung dieselbe Länge haben; innerhalb der Datei ist keine Anzeige der Länge erforderlich.

Satz variabler Länge:
Satz in einer Datei, in der die Sätze unterschiedlich lang sein können. Die Satzlänge muss im ersten Wort innerhalb des Satzes angegeben werden. Dieses Satzlängenfeld wird bei der Ermittlung der Satzlänge mit einbezogen. Das Satzlängenfeld ist 4 Byte lang und enthält die Satzlänge linksbündig als Sedezimal- oder Dezimalzahl.

Satzformat

Festlegung einer Datei hinsichtlich der Länge und Segmentierung ihrer Sätze.

Satzformat V:
Wird ein Band des Satzformats V im Nicht-EBCDI-Code beschrieben, wird die Länge sedezimal angegeben.

Satzformat D:
Wird ein Band bei Satzformat D im ISO-7-Bit-Code beschrieben, wird die Länge dezimal 4-stellig angegeben.

SPOOLOUT

automatischer SPOOLOUT: automatische Ausgabe des Inhalts der Systemdatei SYSLST auf einen Drucker oder Versenden per E-Mail bei Auftragsende (EXIT-JOB bzw. LOGOFF).

Stapelbetrieb

Siehe Batchbetrieb

SYSFILE-Umgebung

Siehe Systemdateien;
Als SYSFILE-Umgebung kann die Gesamtheit der einem Auftrag zugewiesenen Systemdateien bezeichnet werden.

Systemdateien

einem Auftrag zugewiesene System-Ein-/Ausgabedateien.
Die (Standard-) Dateinamen SYSDTA, SYSSTMT, SYSCMD, SYSIPT, SYSLST, SYSLST01, SYSLST02, ..., SYSLST99, SYSOPT und SYSOUT bezeichnen vom Betriebssystem benutzte (System-) Dateien zur Daten- bzw. Kommandoeingabe an das Betriebssystem oder zur Datenausgabe durch das Betriebssystem. Diese Dateien werden jeweils durch die Task eingerichtet und bezeichnen vorgegebene Ein- bzw. Ausgabebereiche.
Der Anwender kann die primäre Zuordnung aufheben und den (Standard-) Dateinamen eigene katalogisierte Dateien bzw. zusammengesetzte S-Variablen (bei Einsatz des Software-Produkts SDF-P) zuweisen.
Ausführliche Informationen zu Systemdateien siehe Handbuch „Kommandos“ [3].

TFT (= Task File Table)

Tabelle, die mit dem Kommando ADD-FILE-LINK, Operand LINK-NAME, erstellt wird und aus der bei Dateieröffnung Datei- und Verarbeitungseigenschaften in den Dateisteuerblock übernommen werden.

TSN (= Task Sequence Number)

Siehe Auftragsnummer (= TSN, Task Sequence Number)

TST (Tape Set Table)

Tabelle, die die für einen Auftrag angeforderten Bänder anzeigt (in Zusammenhang mit TFT).

Übertragungsbetrieb

Im Übertragungsbetrieb gibt der Anwender die Lage des Satzes in seinem Programm an. Für die Datenübertragung zum und vom Puffer ist das System zuständig.

ungeblockter Satz (unblocked record)

Ein Satz in einer Datei, in der jeder Datenblock nur einen Satz oder ein Satzsegment enthalten darf.

VSEQ (= Volume Sequence Number)

Bezeichnet einen Dateiabschnitt bei Mehrbanddateien.

VTOC (= Volume Table of Contents)

Dateiverzeichnis im F1-Etikett einer privaten Platte oder auf einem Net-Storage-Volume.

Zugriffsmethode

Eine festgelegte Technik der Datenverwaltung, die dem Benutzer die Datenorganisation sowie die Methode der Datenübertragung zwischen den Ein-/Ausgabegeräten und dem Arbeitsspeicher vorschreibt.

Zugriffsmethoden des DVS sind:

- EAM (Evanescent Access Method)
- SAM (Sequential Access Method)
- ISAM (Indexed Sequential Access Method)
- UPAM (User Primary Access Method)
- BTAM (Basic Tape Access Method)

26 Literatur

Die Handbücher finden Sie im Internet unter *http://bs2manuals.ts.fujitsu.com*. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie auch in gedruckter Form bestellen.

[1] **BS2000 OSD/BC**
DVS-Makros
Benutzerhandbuch

[2] **BS2000 OSD/BC**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

[3] **BS2000 OSD/BC**
Kommandos
Benutzerhandbuch

[4] **SPOOL** (BS2000)
Benutzerhandbuch

[5] **DAB** (BS2000)
Disk Access Buffer
Benutzerhandbuch

[6] **RFA** (BS2000)
Remote File Access
Benutzerhandbuch

[7] **BS2000 OSD/BC**
Einführung in die Systembetreuung
Benutzerhandbuch

[8] **SECOS** (BS2000)
Security Control System - Zugangs- und Zugriffskontrolle

Benutzerhandbuch

[9] **ARCHIVE** (BS2000)
Benutzerhandbuch

[10] **HSMS** (BS2000)
Hierarchisches Speicher Management System
Benutzerhandbuch

[11] **HIPLEX MSCF** (BS2000)
BS2000-Rechner im Verbund
Benutzerhandbuch

[12] **PERCON** (BS2000)
Benutzerhandbuch

[13] **BS2000 OSD/BC**
Dienstprogramme
Benutzerhandbuch

[14] **DRV** (BS2000)
Dual Recording by Volume
Benutzerhandbuch

[15] **BS2000 OSD/BC**
Systeminstallation
Benutzerhandbuch

[16] **SDF-A** (BS2000)
Benutzerhandbuch

[17] **SDF-P** (BS2000)
Programmieren in der Kommandosprache
Benutzerhandbuch

[18] **BS2000 OSD/BC**
Dateien und Volumes größer 32 GB
Benutzerhandbuch

[19] **RSO** (BS2000)
Remote SPOOL Output
Benutzerhandbuch

[20] **JV** (BS2000)
Jobvariablen
Benutzerhandbuch

[21] **XHCS** (BS2000)
8-bit-Code-Verarbeitung im BS2000
Benutzerhandbuch

[22] **BS2000 OSD/BC**
System Managed Storage
Benutzerhandbuch

[23] **FUJITSU Server BS2000 SE Serie**
Bedienen und Verwalten
Benutzerhandbuch

[24] **CRYPT** (BS2000)
Sicherheit mit Kryptographie
Benutzerhandbuch

[25] **VM2000** (BS2000)
Virtuelles Maschinensystem
Benutzerhandbuch

[26] **SHC-OSD** (BS2000)
Storage Management für BS2000
Benutzerhandbuch