

Deutsch



FUJITSU Software

# BS2000 OSD/BC V11.0

Dateien und Volumes größer 32 GB

Benutzerhandbuch

Ausgabe Juli 2017

## **Kritik... Anregungen... Korrekturen...**

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

Sie können uns Ihre Kommentare per E-Mail an [manuals@ts.fujitsu.com](mailto:manuals@ts.fujitsu.com) senden.

## **Zertifizierte Dokumentation nach DIN EN ISO 9001:2008**

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 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH  
[www.cognitas.de](http://www.cognitas.de)

## **Copyright und Handelsmarken**

Copyright © 2017 Fujitsu Technology Solutions GmbH.

Alle Rechte vorbehalten.

Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.

---

# Inhalt

<b>1</b>	<b>Einleitung</b> . . . . .	<b>7</b>
<b>1.1</b>	<b>Zielgruppen des Handbuchs</b> . . . . .	<b>7</b>
<b>1.2</b>	<b>Konzept des Handbuchs</b> . . . . .	<b>8</b>
<b>1.3</b>	<b>Darstellungsmittel</b> . . . . .	<b>10</b>
<b>2</b>	<b>Große Objekte im BS2000</b> . . . . .	<b>11</b>
<b>2.1</b>	<b>Pubsets</b> . . . . .	<b>12</b>
<b>2.2</b>	<b>Volumes</b> . . . . .	<b>15</b>
<b>2.3</b>	<b>Dateien</b> . . . . .	<b>17</b>
<b>2.4</b>	<b>POSIX-Dateisysteme</b> . . . . .	<b>20</b>
2.4.1	Große POSIX-Dateisysteme . . . . .	20
2.4.2	Große POSIX-Dateien . . . . .	21
<b>2.5</b>	<b>Benutzerprogramme für Konfigurationen mit großen Dateien</b> . . . . .	<b>22</b>
<b>3</b>	<b>Systembetreuer</b> . . . . .	<b>23</b>
<b>3.1</b>	<b>Pubset-Verwaltung</b> . . . . .	<b>24</b>
3.1.1	Einrichten und Erweitern von Large-Objects-Pubsets mit SIR . . . . .	25
3.1.2	Upgrade und Erweiterung existierender Pubsets . . . . .	26
3.1.3	Generieren von SM-Pubsets mit SMPGEN . . . . .	27
3.1.4	Recovery von SM-Pubsets mit großen Objekten . . . . .	27
3.1.5	Large-Objects-Pubsets im Verbund . . . . .	27
<b>3.2</b>	<b>Katalog- und Dateiverwaltung</b> . . . . .	<b>28</b>
3.2.1	Berücksichtigung großer Objekte im Katalogeintrag (CE) . . . . .	28
3.2.2	Zuweisen einer 4-Byte-Extent-Liste zu einer Datei . . . . .	29
3.2.3	Erstellen von SYSEAM-Dateien . . . . .	29
3.2.4	Kataloge anlegen . . . . .	30

<b>3.3</b>	<b>Parameterservice</b>	<b>31</b>
3.3.1	Systemparameter FST32GB	31
<b>3.4</b>	<b>Betroffene Kommandos</b>	<b>32</b>
3.4.1	SET-PUBSET-ATTRIBUTES / SHOW-PUBSET-ATTRIBUTES	33
3.4.2	MODIFY-USER-PUBSET-ATTRIBUTES	35
3.4.3	Nichtprivilegierte Kommandos	35
<b>3.5</b>	<b>Ablauffähigkeit von Dienstprogrammen in Umgebungen mit großen Objekten</b>	<b>36</b>
3.5.1	HSMS/ARCHIVE	36
3.5.2	VOLIN	37
3.5.3	SPCCNTRL	37
<b>3.6</b>	<b>Fehlerquellen und Konflikte</b>	<b>38</b>
3.6.1	Einschränkungen für Systemdateien	38
3.6.2	Einschränkungen für große Volumes	38
<b>4</b>	<b>Anwender und Programmierer</b>	<b>39</b>
<b>4.1</b>	<b>Benutzerkommandos</b>	<b>40</b>
4.1.1	SHOW-MASTER-CATALOG-ENTRY	41
4.1.2	ADD-FILE-LINK / SHOW-FILE-LINK	41
4.1.3	SHOW-FILE-ATTRIBUTES	42
<b>4.2</b>	<b>Assembler-Makros</b>	<b>43</b>
4.2.1	STAMCE	45
4.2.2	FSTAT	49
4.2.3	OPEN	53
4.2.4	FCB	55
4.2.5	FILE	56
4.2.6	RDTFT	57
4.2.7	DIV	58
4.2.8	FPAMSRV	62
4.2.9	FPAMACC	64
4.2.10	Besonderheiten bei SHARUPD=YES	65
<b>4.3</b>	<b>Hinweise zu in höheren Programmiersprachen erstellten Programmen</b>	<b>67</b>
<b>5</b>	<b>Hinweise zur Migration</b>	<b>69</b>
<b>5.1</b>	<b>Vorüberlegungen</b>	<b>69</b>
<b>5.2</b>	<b>Umgebung</b>	<b>70</b>
<b>5.3</b>	<b>Programmumstellung</b>	<b>71</b>

---

5.3.1	Assemblerprogramme . . . . .	71
5.3.2	Programme in höheren Programmiersprachen . . . . .	72
<b>6</b>	<b>Anhang . . . . .</b>	<b>73</b>
<hr/>		
6.1	Semantische Inkompatibilitäten . . . . .	74
6.2	Meldungen bzgl. großer Objekte . . . . .	75
	<b>Literatur . . . . .</b>	<b>79</b>
<hr/>		
	<b>Stichwörter . . . . .</b>	<b>81</b>
<hr/>		



---

# 1 Einleitung

BS2000 OSD/BC unterstützt Dateien und Volumes mit einer Kapazität bis zu 4 Terabyte. Dateien und Volumes, deren Größe 32 GB überschreitet, werden „große Dateien“ und „große Volumes“ genannt und beide auch als „große Objekte“ bezeichnet.

Große Volumes können an allen BS2000-Servern an den über FibreChannel angeschlossenen externen Plattenspeicher-Subsystemen konfiguriert werden.

Große Dateien und große Volumes werden nur in speziellen Pubsets unterstützt, die für die Verwendung dieser großen Objekte attribuiert werden müssen. Diese Pubsets werden auch als „große Pubsets“ bezeichnet und können an allen heute freigegebenen BS2000-Versionen importiert werden.

An bestimmten Datenstrukturen, u.a. im Katalogeintrag, müssen die bisher für die Seiten-Adressierung verwendeten 3-Byte-Felder in 4-Byte-Felder umgesetzt werden.

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.

## 1.1 Zielgruppen des Handbuchs

Dieses Handbuch wendet sich an Programmierer und Systembetreuer. Es soll den Umstieg auf große Dateien und große Volumes erleichtern.

Voraussetzung für die Arbeit mit diesem Handbuch sind sehr gute Kenntnisse des DVS sowie der Kommandosprache SDF (für Systembetreuer) bzw. der Programmiersprache Assembler (für Programmierer).

## 1.2 Konzept des Handbuchs

Im [Kapitel „Große Objekte im BS2000“](#) werden die theoretischen Grundlagen für die Einführung großer Objekte (großer Volumes und großer Dateien) beschrieben. Die Pubsets, Volumes und Dateien betreffenden Änderungen im BS2000 werden aufgezeigt sowie allgemeine Einschränkungen genannt. Für Benutzerprogramme gibt es eine Klassifikation, die den Umgang der einzelnen Programme mit großen Dateien beschreibt.

Das [Kapitel „Systembetreuer“](#) gibt eine Übersicht über die Rolle des Systembetreuers bei der Einführung und Pflege großer Objekte.

Schwerpunkt ist die Erweiterung des Aufgabenbereichs „Pubset-Verwaltung“. Aber auch für die „Katalog- und Dateiverwaltung“ sowie für „Systemeinleitung und Parameterservice“ sind verschiedene Aspekte zu beachten. Ein Abschnitt beschäftigt sich mit den Änderungen an Kommandoschnittstellen bzgl. der Einführung großer Objekte.

Weiter wird auf Anpassungen in einzelnen, für die Systembetreuung wichtigen Dienstprogrammen hingewiesen, die von den Erweiterungen um große Objekte betroffen sind. Den Abschluss des Kapitels bildet die Zusammenfassung von Fehlerquellen und Konflikten.

Im [Kapitel „Anwender und Programmierer“](#) werden die Erweiterungen der nicht-privilegierten Kommandoschnittstellen und der Assembler-Makroschnittstellen bzgl. großer Dateien ausführlich beschrieben. Es folgen Hinweise zu RFA und höheren Programmiersprachen, die von den Erweiterungen um große Objekte betroffen sind. Den Abschluss des Kapitels bildet die Zusammenfassung von Fehlerquellen und Konflikten.

Im [Kapitel „Hinweise zur Migration“](#) werden Hinweise zur Einführung von großen Dateien gegeben. Die dazu notwendigen technischen Erweiterungen sind in den vorangehenden Kapiteln beschrieben worden. Hier soll aufgezeigt werden, in welchen Schritten die Migration erfolgen könnte, und welche Überlegungen in eine entsprechende Planung einfließen sollten.



## 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://manuals.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://manuals.ts.fujitsu.com>.

## 1.3 Darstellungsmittel

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 [Seite 79](#) aufgeführt.

Kommandos, auf die in diesem Handbuch verwiesen wird, sind im Handbuch „Kommandos“[\[10\]](#) beschrieben. Die genannten Makros sind in den Handbüchern „Makroaufrufe an den Ablaufteil“ [\[11\]](#) und „DVS-Makros“ [\[3\]](#) beschrieben. Die Metasyntax der Kommandos bzw. Makros ist in den entsprechenden Handbüchern enthalten.



Mit diesem Piktogramm werden Warnhinweise gekennzeichnet, die auf einen möglichen Datenverlust oder schwerwiegende Konflikte aufmerksam machen sollen.

---

## 2 Große Objekte im BS2000

Große Volumes und große Dateien werden im Folgenden als „große Objekte“ bezeichnet.

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 Maximalwerte:

- Kapazität eines Pubsets oder Volume-Sets: ca. 4 TB (2.147.483.647 PAM-Seiten)
- Kapazität eines Volumes: ca. 2TB (1.073.741.824 PAM-Seiten)
- Dateigröße: ca. 4 TB

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

Blocknummern und Blockzähler sind an verschiedenen Benutzerschnittstellen von BS2000 OSD/BC sichtbar. Während alle neueren Ausprägungen dieser Schnittstellen konsequent 4-Byte-Felder verwenden, werden bei manchen älteren Schnittstellenversionen 3-Byte-Felder verwendet. Hier ist beim Vorhandensein von Dateien  $\geq 32$  GB (und in seltenen Fällen auch bei Volumes  $\geq 32$  GB) mit Kompatibilitätsproblemen zu rechnen.

Dem Kompatibilitätsaspekt wird Rechnung getragen, indem für große Objekte neue Pubset-Typen als spezielle Behälter bereit gestellt werden, wodurch eine Abschottung von der bisherigen Welt der „kleinen Objekte“ erreicht und eine sukzessive Einführung großer Volumes und großer Dateien ermöglicht wird.

Für die Einführung von großen Volumes und großen Dateien gelten jeweils unterschiedliche Randbedingungen:

- Die Einführung von großen Volumes dient dem einfachen, kostengünstigen Ausbau der Kapazitäten im Data Center. Sie erfolgt für Anwender und Programme nicht sichtbar.
- Große Dateien werden benötigt, um das weitere Wachstum von Anwendungen zu ermöglichen, bei denen die Größe einzelner (i.d.R. weniger) Dateien die bisherige Grenze von 32 GB in absehbarer Zeit überschreitet.  
Hierzu ist u.U. eine Anpassung der beteiligten Programme, die z.B. alte Ausprägungen von bestimmten Schnittstellen verwenden oder implizit eine max. Dateigröße von 32 GB annehmen, zwingend erforderlich, damit die Datenintegrität bei Überschreiten der 32-GB-Grenze gewährleistet wird.  
Eine Auswirkung auf Programme, die selbst nicht mit großen Dateien arbeiten sollen, wird ausgeschlossen.

## 2.1 Pubsets

Standard-Pubsets können keine großen Objekte enthalten. Die Aufnahme von großen Objekten ermöglichen nur so genannte Large-Objects-Pubsets, bei denen die folgenden zwei Pubset-Typen unterschieden werden:

- Large-Objects-Pubsets ohne große Dateien:

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

Die Einführung großer Volumes kann nur Auswirkungen auf die PHP der Extent-Listen haben, einer Größe, die für Anwenderprogramme nicht von Interesse ist und keinesfalls durch sie direkt geändert werden kann. Daher erlaubt dieser Pubset-Typ eine (weitgehend) problemlose Integration von großen Volumes in bestehende Anwendungsumgebungen. Diese Pubsets verhalten sich aus Anwendersicht (fast) wie konventionelle Pubsets.

- Large-Objects-Pubsets mit großen Dateien

Bei diesem Pubset-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.

Diese beiden Pubset-Typen sind seit OSD-BC V5.0 bekannt und können in allen aktuell freigegebenen BS2000-Versionen importiert werden.

Große Objekte werden sowohl auf SF- als auch auf SM-Pubsets unterstützt. Dazu existieren zwei zusätzlicher Attribute:

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



Auch bei SM-Pubsets sind die Attribute LARGE\_OBJECTS und LARGE\_FILES\_ALLOWED Eigenschaften des Pubsets und nicht von (einzelnen) Volume-Sets, da andernfalls Volume-Sets an verschiedenen Benutzerschnittstellen sichtbar wären und damit Inkompatibilitäten verursachen könnten.

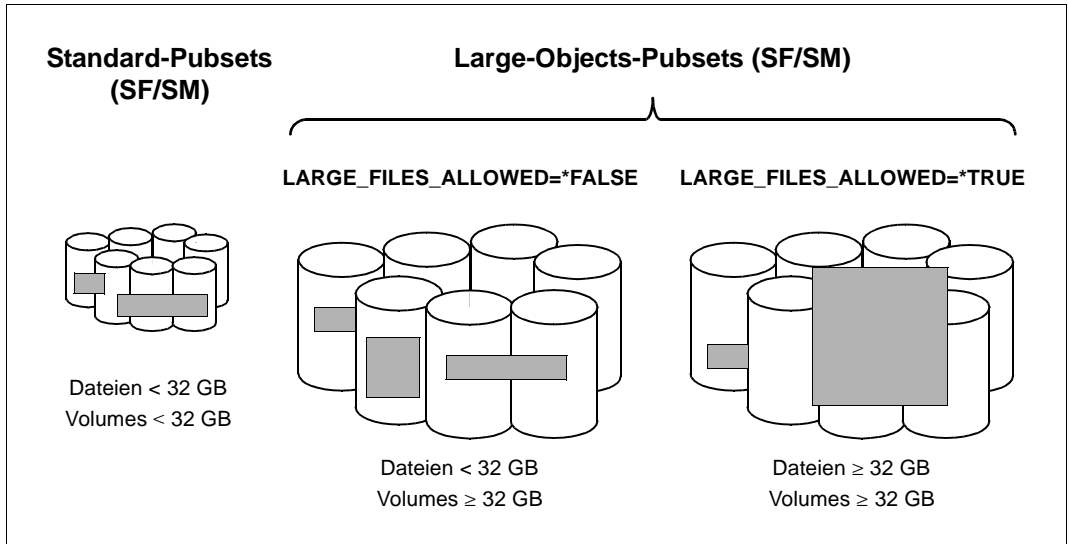
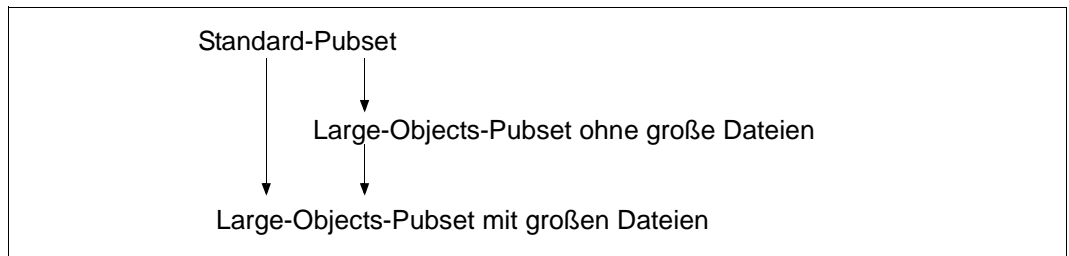


Bild 1: Attribute von Large-Objects-Pubsets

### Definieren von Large-Objects-Pubsets

Die Eigenschaften `LARGE_OBJECTS` und `LARGE_FILES_ALLOWED` sind statische Eigenschaften, die beim Einrichten des Pubsets mit SIR festgelegt werden können (siehe [Seite 25](#)).

Bestehende Standard-Pubsets können mit dem Kommando `SET-PUBSET-ATTRIBUTES` zu Large-Objects-Pubsets mit und ohne Unterstützung großer Dateien hochgestuft werden (siehe [Seite 33](#)). Dabei sind folgende Upgrades möglich:



Beim Import des Pubsets werden dann die Attribute in den MRSCAT übernommen.



### Achtung!

Die Rückkonvertierung in einen Standard-Pubset ist nicht möglich.

Das Vorhandensein großer Dateien kann in bestimmten Fällen die Ablauffähigkeit von Anwendungen beeinträchtigen (siehe dazu [Kapitel „Anwender und Programmierer“](#) ab Seite 39).

### Übersicht über die 32-GB-relevanten Schnittstellen zur Pubset-Verwaltung

Schnittstelle	Änderung
privilegierte Kommandos	
SET-PUBSET-ATTRIBUTES	Festlegung, ob existierende Pubsets zu Large-Objects-Pubsets ohne oder mit großen Dateien hochgestuft werden
SHOW-PUBSET-ATTRIBUTES	Zwei S-Variablen zeigen den Inhalt der Attribute LARGE-VOLUMES und LARGE-FILES an: var(*LIST).LARGE-VOL und var(*LIST).LARGE-FILE
SHOW-MASTER-CATALOG-ENTRY	Ausgabe der LARGE_OBJECTS-Eigenschaften bei Large-Objects-Pubsets
Makros	
STAMCE	Ausgabe von MRSCAT-Einträgen bzgl. LARGE_OBJECTS
privilegierte Dienstprogramme	
SIR	Installieren und Erweitern von Pubsets, Initialisieren von Volumes bzgl. LARGE_OBJECTS

Tabelle 1: Übersicht über die 32-GB-relevanten Schnittstellen zur Pubset-Verwaltung

## 2.2 Volumes

Analog zu Pubsets besitzen große Volumes ein eigenes Attribut: LARGE\_VOLUME.

Das Attribut LARGE\_VOLUME charakterisiert ein logisches Volume und zeigt an, dass das Volume eine Bruttokapazität von  $\geq 32$  GB hat.

Das Attribut LARGE\_VOLUME wird beim Initialisieren des Volumes mit VOLIN im Basis-Record des SVL abgelegt.

Wie die Eigenschaft LARGE\_OBJECTS für Pubsets ist die Eigenschaft LARGE\_VOLUME für Volumes dauerhaft und wird im SVL abgelegt.

### Einschränkungen für große Volumes

- Große Volumes können nur in Large-Objects-Pubsets aufgenommen werden.
- VOLIN  
Für beliebige Formatierung beherrscht VOLIN als maximale Kapazität 2 TB.
- Als Privatplatten sind große Volumes nicht zulässig. Der Versuch, ein großes Volume als Privatplatte zu initialisieren, wird mit der folgenden Fehlermeldung abgewiesen:

```
NVL0146  PRIVATPLATTEN MIT EINER KAPAZITAET GROESSER/GLEICH 32 GB SIND
        NICHT ZULAESSIG
```

### 32-GB-relevante Schnittstelle zur Volume-Verwaltung

Schnittstelle	Änderung
privilegierte Dienstprogramme	
SIR	Installieren und Erweitern von Pubsets, Initialisieren von Volumes bzgl. LARGE_OBJECTS

Tabelle 2: Übersicht über die 32-GB-relevanten Schnittstellen zur Volume-Verwaltung

### **Performante Bedienung großer Volumes**

Wenn beim Einsatz großer Volumes die Last auf dem Volume entsprechend der größeren Datenmenge zunimmt, so sollten Vorkehrungen für ihre performante Bedienung getroffen werden:

- Zur Performance-Steigerung (und Erhöhung der Datensicherheit) können externe Plattenspeicher-Subsysteme mit der Festplattentechnik RAID betrieben werden. Beim Einsatz großer Volumes wird RAID 5, RAID 6 oder besser RAID 1/0 empfohlen. Dabei werden die Daten eines logischen Volumes über mehrere physikalische Platten verteilt (Striping).
- Zusätzlich wird auf /390-Servern der Einsatz der BS2000-Funktion PAV (Parallel Access Volume) empfohlen um die Platten-Ein-/Ausgaben zu parallelisieren. Auf x86-Servern ist die analoge Funktion mittels RSC für Platten-I/Os standardmäßig vorhanden.

Die Kombination beider Maßnahmen führt zu erheblichen Verbesserungen des TP- und Batch-Betriebs (bei Mehr-Task-Batch), sowohl hinsichtlich der I/O-Zeiten als auch bezüglich Durchsatz. Siehe auch „Performance Handbuch [12].



## 2.3 Dateien

Die maximale (theoretische) Dateigröße beträgt ca. 4 TB (2.147.483.647 PAM-Seiten). Dies bedeutet, dass innerhalb des Betriebssystems konsequent 4-Byte-Blocknummern und 4-Byte-Zähler für Datei- und Plattengrößen verwendet werden müssen (siehe [Bild 2](#)). (Die Bezeichnung „Block“ wird hier synonym mit den Bezeichnungen „PAM-Seite“ oder „Half Page“ verwendet.)

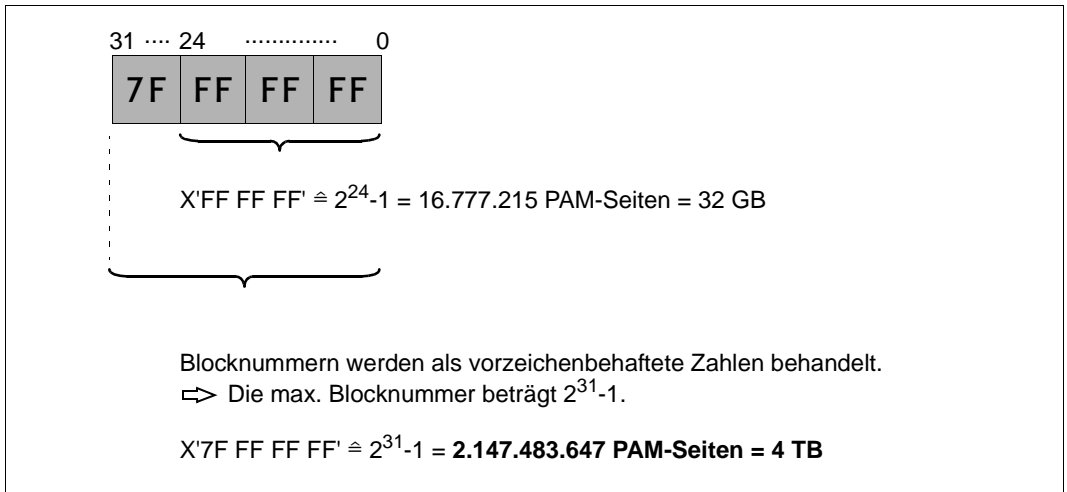


Bild 2: 3-Byte- und 4-Byte-Felder und daraus resultierende Datei- und Datenträgergrößen

### Erweiterung des Katalogeintrags

Zentral für die Aufhebung der 32-GB-Grenze für die Volume- und Dateigröße war 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.

#### 3-Byte- und 4-Byte-Felder

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  $\geq 32$  GB vorhanden, ist mit Kompatibilitätsproblemen zu rechnen; in seltenen Fällen auch, wenn „nur“ Volumes  $\geq 32$  GB vorhanden sind.

### Zusätzliches Format für die Extent-Liste

Mit der Einführung von 4-Byte-LHPs und 4-Byte-PHPs musste für die Extent-Liste ein zusätzliches Format eingeführt werden.

Der Zusammenhang zwischen der maximalen Größe von Datenträgern und Dateien und der Feldbreite von LHP und PHP wird in [Bild 3](#) dargestellt:

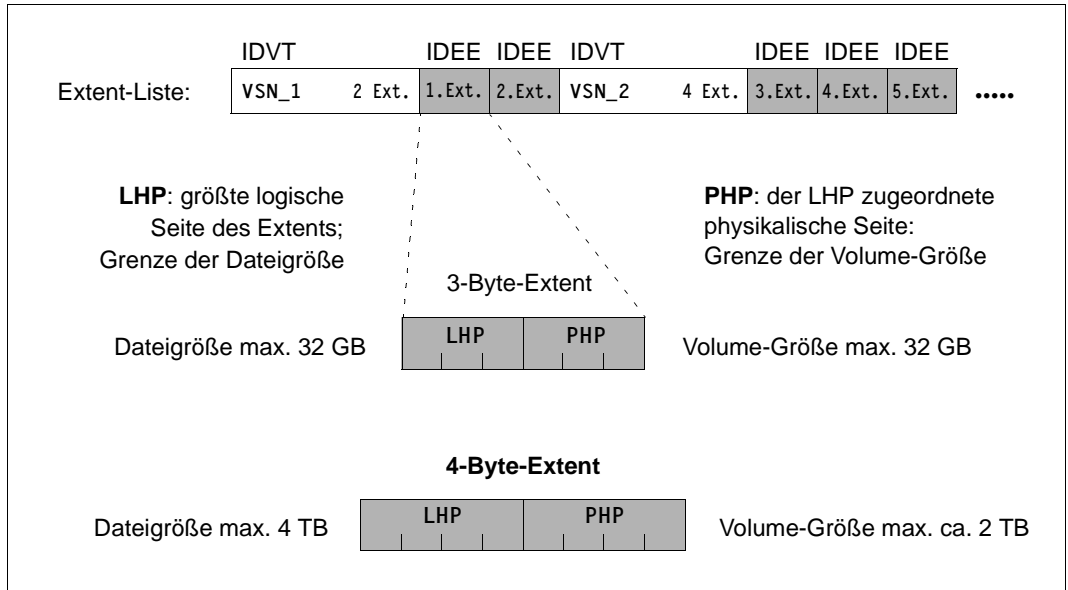


Bild 3: Datei- und Datenträgergröße im Zusammenhang mit der Feldbreite von LHP und PHP

Es werden beide Formate der Extent-Liste unterstützt:

- Grundsätzlich wird das „alte“ Format mit 3-Byte-Blocknummern verwendet.
- Nur im Fall von großen Dateien oder von Dateien, die ganz oder teilweise auf großen Volumes liegen, 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

- Die Paging-Datei darf keine Datei  $\geq 2$  TB sein.
- Eine SYSEAM-Datei darf keine Datei  $\geq 32$  GB sein.
- Dateien mit BLKCTRL=PAMKEY werden nicht unterstützt, da im Systemteil des Pamkey die logische Seitennummer als 3-Byte-Feld hinterlegt ist.

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

Schnittstelle	Änderung
privilegierte Kommandos	
SET-PUBSET-ATTRIBUTES	Festlegung, ob existierende Pubsets zu Large-Objects-Pubsets ohne oder mit großen Dateien hochgestuft werden
SHOW-PUBSET-ATTRIBUTES	Zwei zusätzliche S-Variablen zeigen den Inhalt der Attribute LARGE-VOLUMES und LARGE-FILES an: var(*LIST).LARGE-VOL und var(*LIST).LARGE-FILE
SHOW-MASTER-CATALOG-ENTRY	Ausgabe der LARGE_OBJECTS-Eigenschaften bei Large-Objects-Pubsets
nicht-privilegierte Kommandos	
ADD-FILE-LINK	zusätzlicher Operand für große Dateien
SHOW-FILE-LINK	Ausgabe des Attributs „Datei darf groß werden“
SHOW-FILE-ATTRIBUTES	erweiterte Ausgabestellen verschiedener Ausgabefelder
Makros	
FCB	zusätzlicher Operand für große Dateien
FILE	zusätzlicher Operand für große Dateien
FSTAT	Prüfungs- und Umstellungsaufwand bei VERSION=0/1
OPEN	Semantikproblem beachten
RDTFT	Ausgabe des Attributs „Datei darf groß werden“
DIV	zusätzlicher Operand für große Dateien; vergrößerter Wertebereich für BLOCK und SPAN
FPAMACC	vergrößerter Wertebereich für BLOCK
FPMSRV	zusätzlicher Operand für große Dateien
STAMCE	Ausgabe von MRSCAT-Einträgen bzgl. LARGE_OBJECTS

Tabelle 3: Übersicht über die 32-GB-relevanten Schnittstellen zur Datei-Verwaltung

## 2.4 POSIX-Dateisysteme

Die 32-GB-Grenze des BS2000 hat innerhalb von POSIX keine Bedeutung. In der Vergangenheit hatte POSIX einstmals jedoch nur solche Dateien unterstützt, die maximal 2 Gigabyte groß sein können. Dies lag daran, dass die Daten innerhalb einer Datei mit einer Variablen des Datentyps integer (vorzeichenbehaftet) adressiert wurden. Damit lassen sich maximal  $2^{31}-1$  Bytes adressieren, das sind 2 Gigabyte.

Diese Grenze verursachte bei verschiedenen Einsatzfällen Probleme, z.B. bei Druckdateien mit speicherintensiven Grafiken, weshalb mehr Anwender forderten, die maximale Dateigröße zu erhöhen. Dies wurde auch von den Standardisierungsgremien unterstützt und führte dazu, dass ein neuer Standard für große Dateien festgelegt wurde.

### Standard für große Dateien

Kernpunkt dieses Standards ist, dass für die Adressierung innerhalb einer Datei eine Variable des Datentyps long long verwendet wird. Dieser Datentyp besteht aus einem integer-Paar, so dass man inklusive Vorzeichen eine Adressbreite von  $2^{63}-1$  Bytes erreicht.

Diese neue Klasse von Dateien sollte natürlich möglichst kompatibel zu den bestehenden sein, damit existierende Programme ohne großen Aufwand auch mit großen Dateien arbeiten können. D.h. große Dateien sollten syntaktisch und semantisch möglichst mit denselben Schnittstellen bearbeitet werden können wie bisherige Dateien.

### 2.4.1 Große POSIX-Dateisysteme

Die maximale Größe einer POSIX-Datei ist auch durch die Größe des Dateisystems beschränkt, in dem diese Datei liegt, d.h durch die Größe der Behälterdatei in BS2000.

Da die interne Adressierung jetzt Variablen vom Typ long long verwendet, lässt sich ein viel größerer Bereich adressieren, sodass Behälterdateien und damit auch die POSIX-Dateisysteme maximal 1TB werden können.

Die Differenz zwischen maximaler Größe von BS2000-Dateien und maximaler Behältergröße ist durch die vorgegebene Implementierung in POSIX bestimmt.

Die Größe einer Behälterdatei und damit eines POSIX-Dateisystems wird beim Einrichten mit dem Verwaltungs-Tool POSINST festgelegt (siehe Handbuch „POSIX Grundlagen“ [13]).

## 2.4.2 Große POSIX-Dateien

Große POSIX-Dateien sind Dateien eines POSIX-Dateisystems, die größer als 2 Gigabyte sein können. Große POSIX-Dateien können nur in POSIX-Dateisystemen angelegt werden, die auf einem großen Behälter basieren und damit den Grenzwert von 2 Gigabyte überschreiten können, siehe vorheriger Abschnitt.

Die maximale Größe einer POSIX-Datei ist durch die Größe der Behälterdatei begrenzt, die sie enthält. Außerdem können Sie in POSIX eine maximale Dateigröße angeben, die für alle Dateien des POSIX-Dateisystems gilt (Kommando *ulimit* oder Parameter FILESIZE in der POSIX-Informationsdatei).

### Programmschnittstellen für große POSIX-Dateien

Um mit POSIX-Dateien zu arbeiten, gibt es eine Anzahl von C-Bibliotheksfunktionen, wie *open()*, *close()*, die von CRTE zur Verfügung gestellt werden. Eine Teilmenge dieser Funktionen liegt in 64-bit-Ausprägung vor, um große POSIX-Dateien zu bearbeiten. Diese Funktionen haben den gleichen Namen, ergänzt um den Suffix „64“, z.B. *open64()*. Außerdem wurden einige Datenstrukturen und -typen auf 64-bit umgestellt.

Näheres zur Programmschnittstelle siehe Handbuch „POSIX Grundlagen“ [13].

### Shell-Kommandos für große POSIX-Dateien

Die meisten Dateiverarbeitungs-Kommandos der POSIX-Shell können große POSIX-Dateien erkennen und z.T. auch verarbeiten. Dabei unterscheidet man 2 Kategorien:

**large file aware** Das Kommando kann große POSIX-Dateien korrekt bearbeiten. Einige der Kommandos dieser Kategorie können große Dateien nur bis zu einer bestimmten Dateigröße bearbeiten, zum Beispiel *cpio* bis max. 8 GB.

**large file safe** Das Kommando erkennt große POSIX-Dateien, weist die Verarbeitung jedoch zurück, z.B. mit einer entsprechenden Meldung.

Näheres zur Kommandoschnittstelle siehe Handbuch „POSIX Grundlagen“ [13].

## 2.5 Benutzerprogramme für Konfigurationen mit großen Dateien

Wie bereits erwähnt, kann man nicht davon ausgehen, 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. Große Platten brauchen nicht betrachtet zu werden, da es, wie bereits erwähnt, keine Benutzerschnittstellen dafür gibt. 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 nicht auf die Bearbeitung großer Dateien und/oder ihrer Metadaten vorbereitet. Es ist aber in der Lage, entsprechende – als fehlerhaft zu betrachtende – Zugriffe 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-Kompatibilität zu sehen. Das Wachstum über 32 GB hinaus dürfte sich auf relativ wenige Dateien beschränken; nur Programme, die auf diese zugreifen, müssen LARGE-FILES-fähig sein.

---

## 3 Systembetreuer

Dieses Kapitel gibt eine Übersicht über die Rolle des Systembetreuers bei der Einführung und Pflege großer Objekte.

Schwerpunkt ist die Erweiterung des Aufgabenbereichs „Pubset-Verwaltung“. Aber auch für die „Katalog- und Dateiverwaltung“ sowie für den „Parameterservice“ sind verschiedene Aspekte zu beachten.

Ein Abschnitt beschäftigt sich mit den Änderungen an Kommandoschnittstellen bzgl. der Einführung großer Objekte.

Weiter wird auf Anpassungen in einzelnen, für die Systembetreuung wichtigen Dienstprogrammen hingewiesen, die von den Erweiterungen um große Objekte betroffen sind.

Den Abschluss des Kapitels bildet die Zusammenfassung von Fehlerquellen und Konflikten.

## 3.1 Pubset-Verwaltung

Der Systembetreuer ist für die Einrichtung, Rekonfiguration und Pflege von Pubsets – also auch von Large-Objects-Pubsets – verantwortlich. Zu diesem Zweck stehen ihm als Werkzeuge das Dienstprogramm SIR (siehe [Seite 25f](#)) sowie die Kommandos der Pubset-Verwaltung (siehe [Seite 33ff](#)) zur Verfügung.

Ein Pubset, der große Objekte aufnehmen kann, wird Large-Objects-Pubset genannt. Sowohl SF- als auch SM-Pubsets können die Eigenschaft `LARGE_OBJECTS` besitzen. Bei SM-Pubsets gilt, dass `LARGE_OBJECTS` eine Eigenschaft des gesamten Pubsets, nicht die eines oder mehrerer Volume-Sets ist.

Die maximale Größe eines Large-Objects-Pubsets ist 2.147.483.647 PAM-Seiten oder 4 TB. Diese Begrenzung resultiert aus der derzeitigen Feldbreite von Zählern im Benutzerkatalog und intern geführten Zählern zur Verwaltung von Plattenspeicher.



### Achtung!

Das Vorhandensein großer Dateien kann in bestimmten Fällen Einwirkungen auf die Ablauffähigkeit von Anwendungen haben (siehe dazu [Abschnitt „Ablauffähigkeit von Dienstprogrammen in Umgebungen mit großen Objekten“ auf Seite 36](#), und [Abschnitt „Hinweise zu in höheren Programmiersprachen erstellten Programmen“ auf Seite 67](#)).

Bevor ein Pubset also so attribuiert wird, dass er die Unterstützung von großen Dateien erlaubt, sollte sichergestellt werden, dass die Softwarekonfiguration entsprechend abgestimmt ist. Entsprechende Klärungen und evtl. Anpassungen sollten deshalb im Vorfeld erfolgen.



### 3.1.1 Einrichten und Erweitern von Large-Objects-Pubsets mit SIR

Für SF- und SM-Pubsets können Volumes und Dateien  $\geq 32$  GB zugelassen werden. Diese Pubsets werden damit zu Large-Objects-Pubsets.

#### Kennzeichnen der Zulässigkeit von großen Objekten in einem Pubset

Um große Volumes und/oder Dateien zu erlauben, stellt die SIR-Anweisung DECLARE-PUBSET zum Einrichten und Erweitern von Pubsets zwei Unteroperanden zur Verfügung:

```
//DECLARE-PUBSET PUBSET-TYPE=...(...,ACTION=...(...,
LARGE-DISKS-ALLOWED=*NO/*YES(LARGE-FILES-ALLOWED=*NO/*YES))
```

Der Operand LARGE-DISKS-ALLOWED=\*NO/\*YES bestimmt, ob Volumes mit einer Kapazität  $\geq 32$  GB in ein Volume-Set des Pubsets aufgenommen werden dürfen oder nicht. Die Eigenschaft „große Volumes“ ist unabhängig davon, ob große Dateien erlaubt sein sollen oder nicht.

Der Operand LARGE-FILES-ALLOWED=\*NO/\*YES bestimmt, ob Dateien in dem Pubset die Größe von  $\geq 32$  GB haben dürfen oder nicht. Große Dateien dürfen nur auf Pubsets abgelegt werden, die große Volumes unterstützen (LARGE-DISKS-ALLOWED=\*YES).

Wird mit diesen Operanden keine explizite Erlaubnis für große Volumes und große Dateien gegeben, richtet SIR einen Pubset ein, der keine großen Volumes/Dateien zulässt (\*NO ist jeweils der Standardwert), und generiert damit ein Pubset-Format, das auch zu früheren Versionen kompatibel ist.



#### Achtung!

Die Erlaubnis zu großen Volumes und Dateien kann einem Pubset nicht wieder entzogen werden.

Die Änderung der LARGE\_OBJECTS-Attribute wird erst nach der nächsten Inbetriebnahme des Pubsets (mit dem Kommando IMPORT-PUBSET) wirksam.

#### Beheben von Inkonsistenzen

Wird beim Einrichten eines Pubsets (ACTION=\*INSTALL) ein großes Volume angegeben, obwohl der Operand LARGE-DISKS-ALLOWED=\*YES nicht angegeben wurde, wird das Volume mit der Meldung SIR0308 abgewiesen:

```
SIR0308 PLATTE FUER VSN '(&00)' BESITZT KAPAZITAET > 32GB;
PLATTE ZURUECKGEWIESEN
```

Soll dieses große Volume trotzdem in diesen Pubset eingebunden werden, muss nachträglich der Operand LARGE-DISKS-ALLOWED=\*YES angegeben werden.

Wird beim Erweitern eines Pubsets (ACTION=\*EXTEND) ein großes Volume angegeben, obwohl der Operand LARGE-DISKS-ALLOWED=\*YES nicht angegeben wurde, wird das Volume mit der Meldung SIR0308 (siehe oben) abgewiesen.

Soll dieses große Volume trotzdem in diesen Pubset eingebunden werden, muss die Pubset-Eigenschaft zunächst mit dem Kommando SET-PUBSET-ATTRIBUTES ...,LARGE-VOLUMES=\*ALLOWED geändert werden, bevor SIR das Volume hinzufügen kann.

### 3.1.2 Upgrade und Erweiterung existierender Pubsets

#### Kennzeichnen der Zulässigkeit von großen Objekten in einem Pubset

Existierende Pubsets können mit dem Kommando SET-PUBSET-ATTRIBUTES zu großen Pubsets mit und ohne großen Dateien hochgestuft werden (siehe [Seite 33](#)).

Die Änderung der LARGE\_OBJECTS-Attribute wird erst nach der nächsten Inbetriebnahme des Pubsets (mit dem Kommando IMPORT-PUBSET) wirksam.

#### Hinzufügen von großen Volumes zu einem Pubset

Mit dem Kommando MODIFY-PUBSET-PROCESSING können einzelne Volumes zu einem SF-Pubset oder zu einem Volume-Set eines SM-Pubsets hinzugefügt werden; ebenso können damit neue Volume-Sets zu einem SM-Pubset hinzugefügt werden.

Handelt es sich bei den neu hinzukommenden Volumes um große Volumes oder enthalten die hinzuzufügenden (leeren) Volume-Sets große Volumes, so muss der zu erweiternde Pubset ein Large-Objects-Pubset sein. Andernfalls wird die Erweiterung mit der Meldung DMS1383 (mit Insert '06') abgelehnt:

```
DMS1383  VOLUME INKONSISTENT. FEHLER-TYP '(&00)'. KOMMANDO ABGEWIESEN
         '06': Ein Volume ist groesser als 32 GB, aber der Pubset hat
              nicht das Attribut fuer grosse Volumes.
```

Soll im Rahmen der IMPORT-Verarbeitung ein großes Volume hinzugefügt werden, so gelingt dies ebenfalls nur, wenn es sich bei dem zu erweiternden Pubset um einen Large-Objects-Pubset handelt. Andernfalls wird die Erweiterung mit der Meldung DMS037E (mit Insert &00='04') abgebrochen:

```
DMS037E  FORMAT VON VOLUME '(&02)' NICHT KONSISTENT ZU PUBSET/VOLUMESET
         '(&00)' IMPORT-PUBSET MIT FORMAT-FEHLER '(&01)' ABGEBROCHEN.
         '04': Ein Volume ist groesser als 32 GB, aber der Pubset hat
              nicht das Attribut fuer grosse Volumes.
```

### 3.1.3 Generieren von SM-Pubsets mit SMPGEN

Mit dem Dienstprogramm SMPGEN kann aus mehreren SF-Pubsets ein SM-Pubset generiert werden. Befinden sich unter den SF-Pubsets Large-Objects-Pubsets, so wird auch der resultierende SM-Pubset ein Large-Objects-Pubset.

Befindet sich unter den zusammenzuführenden SF-Pubsets wenigstens einer mit der Eigenschaft `LARGE_FILES_ALLOWED`, dann entsteht ein Large-Objects-Pubset mit der Eigenschaft `LARGE_FILES_ALLOWED`.

### 3.1.4 Recovery von SM-Pubsets mit großen Objekten

Bei der Recovery von SM-Pubsets mit nicht mehr funktionsfähigem Control-Volume-Set muss auch die Information, ob es sich um einen Large-Objects-Pubset handelt, rekonstruiert werden.

Die `LARGE_OBJECTS`-Attribute werden daher nicht nur im SVL der Volres des Control-Volume-Sets hinterlegt, sondern auch in den SVLs der Volres der restlichen Volume-Sets des SM-Pubsets.

Zur Recovery wird die Funktion „Exklusiver Import eines Pubsets mit Konvertierung eines Volume-Sets“ angeboten.

Dabei wird mit einem speziellen IMCAT aus einem Volume-Set ein SF-Pubset erzeugt, sodass ein Zugriff auf die auf dem Volume-Set abgelegten Daten unabhängig von der Funktionsfähigkeit des Control-Volume-Sets möglich wird. Bei der Konvertierung erbt der SF-Pubset die `LARGE_OBJECTS`-Eigenschaften des SM-Pubsets.

Bei einer anschließenden Zusammenführung der aus den Volume-Sets gewonnenen SF-Pubsets zu einem SM-Pubset mit SMPGEN werden die `LARGE_OBJECTS`-Eigenschaften wie im [Abschnitt „Generieren von SM-Pubsets mit SMPGEN“ auf Seite 27](#) beschrieben übernommen.

### 3.1.5 Large-Objects-Pubsets im Verbund

Large-Objects-Pubsets können selbstverständlich auch als Shared-Pubsets betrieben werden.

## 3.2 Katalog- und Dateiverwaltung

Mit der Unterstützung großer Objekte wurde eine neue Version des Katalogeintrags eingeführt. Die Erweiterungen wurden z.T. source-inkompatibel eingeführt, um sicherzustellen, dass Komponenten, die neu produziert werden, die Änderungen für „Große Objekte“ berücksichtigen. Da die Erweiterung des Katalogeintrags und der zugehörigen Zugriffsfunktion für die Programmierung von System-Exits von Interesse sein können, folgt an dieser Stelle eine kurze Zusammenfassung der Änderungen.

### 3.2.1 Berücksichtigung großer Objekte im Katalogeintrag (CE)

Es gilt Folgendes:

- Das Versionskennzeichen muss X'FF03' oder höher sein.
- Anhand des Indikators (Bit) zur Kennzeichnung von Katalogeinträgen mit großen Objekten (große Extent-Liste) wird entschieden, ob das 3-Byte- oder das 4-Byte-Format verwendet wird.
- Die Beschreibung der Extent-Liste für große Objekte erfolgt im 4-Byte-Format; eine Extent-Liste, die nicht zu großen Objekten gehört, bleibt – unter anderem Namen – im 3-Byte-Format:  
Es gibt eine zusätzliche Unterstruktur für die Extent-Liste (DCAEE4 bzw. DCAEE4I), in welcher die Felder für die LHP und die PHP jeweils 4 Byte lang sind. Die alte Unterstruktur für die alte Extent-Listen-Form (IDEE bzw. DCMIDEET) mit den 3-Byte-Feldern bleibt erhalten.
- Für die File-Size im Main-Part des Katalogeintrags gibt es ein zusätzliches Feld in der Länge 4 Byte, gleichzeitig bleibt das alte 3-Byte-Feld im Old-Part (unter anderem Namen) erhalten.
- Die Maximalgröße für Extent-Listen wurde auf 2496 Bytes erweitert.  
Für große Dateien wird die Maximalgröße für Extent-Listen auf 2496 Bytes erweitert. Eine große Datei kann damit wie eine kleine Datei maximal 310 Extents enthalten.
- Für den Last-Halpage-Pointer gibt es ein zusätzliches Feld im Main-Part des Katalogeintrags in der Länge 4 Byte eingeführt; gleichzeitig bleibt das alte 3-Byte-Feld im Old-Part (unter anderem Namen) erhalten.

### 3.2.2 Zuweisen einer 4-Byte-Extent-Liste zu einer Datei

Innerhalb der Large-Objects-Pubsets gibt es verschiedene Ausprägungen bezüglich der Struktur des Katalogeintrags, speziell der Extent-Liste (3-Byte- und 4-Byte-Variante). Der Anwender hat keinen direkten Einfluss darauf, welche Variante einer Datei zugeordnet wird. Dies geschieht implizit über das DVS, wobei eine Datei eine 4-Byte-Extent-Liste erhält, wenn:

- die Größe der Datei die 32-GB-Grenze überschreitet oder
- der Datei vom Allocator ein Extent zugewiesen wird, das auf einem Volume mit Kapazität  $\geq 32$  GB liegt

Eine Rückkonvertierung von 4-Byte-Extent-Listen findet nicht statt, auch wenn die Datei wieder kleiner als 32 GB wird oder keinen Extent mehr auf einem großen Volume hat.

### 3.2.3 Erstellen von SYSEAM-Dateien

Die Dateigröße von SYSEAM-Dateien muss immer  $< 32$  GB sein.

Die Größe von SYSEAM-Dateien wird bestimmt durch Angaben im MRSCAT-Eintrag. Sie wird beim Erzeugen des MRSCAT-Eintrags durch das Kommando `ADD-MASTER-CATALOG-ENTRY EAM=*PAR(...)` festgelegt bzw. kann bei bestehendem Eintrag mit dem Kommando `MODIFY-MASTER-CATALOG-ENTRY EAM=*PAR(...)` geändert werden.

Wenn in einem MRSCAT-Eintrag keine Angaben bzgl. EAM vorliegen, greifen entsprechende Systemparameter. Dabei ist durch die maximal möglichen Größen-Angaben (64512 Units) gewährleistet, dass eine SYSEAM-Datei stets kleiner 32 GB bleibt. Darüber hinaus wird bei einer evtl. notwendigen Primary-/Secondary-Allokierung innerhalb von EAM selbst durch entsprechende Parametrisierung sichergestellt, dass eine SYSEAM-Datei nicht größer als 32 GB werden kann.

Die einzige Möglichkeit, dass eine SYSEAM-Datei die kritische Grenze von 32 GB überschreiten kann, ist deren explizites Einrichten durch den Systembetreuer mit `FILE` bzw. `CREATE-FILE` und entsprechender Dateigröße.

Der erste EAM-Zugriff auf diese Datei wird jedoch abgewiesen, wobei die Meldung `DMS0854` auf der Konsole ausgegeben wird:

```
DMS0854    DIE GROESSE DER SYSEAM-DATEI (&00) DARF 32 GIGABYTES NICHT
           UEBERSCHREITEN
```

Die Meldung beinhaltet als Maßnahme das Löschen der entsprechenden SYSEAM-Datei und deren Neu-Anlage mit entsprechend unkritischer Dateigröße durch den Systembetreuer.

### 3.2.4 Kataloge anlegen

Ab BS2000 OSD/BC V11.0 wird der Katalog immer im Format EXTRA LARGE angelegt. Der Katalog wird dadurch vergrößert und somit die Kapazität bezüglich der Anzahl von Datei- oder JV-Einträgen deutlich erhöht.

Datei-Kataloge von Pubsets aus niedrigeren OSD-Versionen, deren Format NORMAL oder LARGE ist, werden beim IMPORT-PUBSET automatisch in das Format EXTRA-LARGE konvertiert.

Mit SIR ab V20.0 werden Kataloge generell im Format EXTRA-LARGE angelegt.

Das aktuelle Katalogformat kann mit dem Kommando SHOW-PUBSET-CATALOG-ALLOCATION abgefragt werden.

## 3.3 Parameterservice

### 3.3.1 Systemparameter FST32GB

Von den Änderungen zur Unterstützung großer Objekte können auch (Kunden-)Anwendungen betroffen sein, die nicht auf die kritischen Datenfelder FILE-SIZE und HIGHEST-USED-PAGE zugreifen. Als Migrationshilfe wird der Systemparameter FST32GB angeboten.

FST32GB hat nur Einfluss auf folgende FSTAT-Schnittstellen:

- Version=0 (entspricht Version=710) bei Angabe eines vollqualifizierten Dateinamens (jedoch nicht bei Angabe einer Dateigenerationsgruppe mit GEN=YES)
- Version=1 (entspricht Version=800), bei der nicht der Operand FNAM spezifiziert wurde

Zu FSTAT siehe [Seite 49ff.](#)

#### **FST32GB = 0 (Standardeinstellung)**

Befindet sich in der Menge der mit FSTAT selektierten Dateien mindestens eine Datei  $\geq 32$  GB, wird der FSTAT-Aufruf mit dem Returncode X'00000576' zurückgewiesen.

#### **FST32GB = 1 (Überlauf ignorieren)**

Befindet sich in der Menge der mit FSTAT selektierten Dateien mindestens eine Datei  $\geq 32$  GB, wird ein Überlauf der 3-Byte-Datenfelder der PHP in der Extent-Liste generell toleriert und führt nicht zu einer Fehleranzeige. Im Falle eines Überlaufs wird den nicht darstellbaren Datenfeldern der Wert X'FFFFFF' zugewiesen.

#### *Hinweis*

Der Systemparameter FST32GB wird nicht ausgewertet, wenn an der FSTAT-Programmschnittstelle der Indikator LARGE\_PUBSET\_ACCESS=YES (siehe [Seite 52](#)) gesetzt ist.

### 3.4 Betroffene Kommandos

Dieser Abschnitt beschreibt die Kommandoschnittstellen, die von großen Objekten betroffen sind.

Kommando / Operand	Erläuterung
privilegierte Kommandos	
SET-PUBSET-ATTRIBUTES LARGE-VOLUMES= *UNCHANGED/*ALLOWED( LARGE-FILES= *UNCHANGED/*ALLOWED)	Legt fest, ob existierende Pubsets zu Large-Objects-Pubsets ohne oder mit großen Dateien hochgestuft werden.
SHOW-PUBSET-ATTRIBUTES	Die zwei S-Variablen var(*LIST).LARGE-VOL und var(*LIST).LARGE-FILE zeigen den Inhalt der Attribute LARGE-VOLUMES und LARGE-FILES an.
MODIFY-USER-PUBSET-ATTRIBUTES TOTAL-SPACE=*UNLIMITED	Operandenwert, der erlaubt, dass das Speicherplatzkontingent des Benutzers für permanenten Speicherplatz, für temporären Speicherplatz oder für Arbeitsdateien 4 TB überschreiten darf (siehe PERM-, TEMP- bzw. WORKSPACE-LIMITS=*PARAMETERS(...)).
SHOW-PUBSET-CATALOG-ALLOCATION	Kommando, das Informationen über Katalogformat, Belegung und Erweiterbarkeit der Kataloge anzeigt.
nicht-privilegierte Kommandos	
ADD-FILE-LINK EXCEED-32GB=*BY-PROGRAM/ *FORBIDDEN/*ALLOWED	Legt fest, ob die Dateigröße 32 GB überschreiten darf.
SHOW-FILE-LINK	Die S-Variable var(*LIST).EXC-32GB zeigt den Inhalt des Attributs EXCEED-32GB an.
SHOW-FILE-ATTRIBUTES	Wenn die Summe der reservierten PAM-Seiten 32 GB überschreitet, erfolgt die Anzeige in Tausend PAM-Seiten. Zwei zusätzliche S-Variablen enthalten die Anzahl in Tausend PAM-Seiten, wenn die ursprüngliche S-Variable 2147483647 PAM-Seiten erreicht hat: var(*LIST).MIGRATE-S1-RESERVED-T und var(*LIST).PUBSET-RESERVED-T.
SHOW-MASTER-CATALOG-ENTRY	Bei Large-Objects-Pubsets werden die LARGE_OBJECTS-Eigenschaften ausgegeben.

Tabelle 4: Kommandos - Übersicht



### 3.4.1 SET-PUBSET-ATTRIBUTES / SHOW-PUBSET-ATTRIBUTES

#### Kennzeichnung der Zulässigkeit von großen Objekten in einem Pubset

Existierende Pubsets können mit dem Kommando SET-PUBSET-ATTRIBUTES zu großen Pubsets hochgestuft werden.

Die folgenden zwei Operanden bestimmen die Zulässigkeit von großen Objekten:

```
/SET-PUBSET-ATTRIBUTES . . . ,LARGE-VOLUMES=*UNCHANGED/*ALLOWED(  
LARGE-FILES=*UNCHANGED/*ALLOWED)
```

Der Operand LARGE-VOLUMES bestimmt, ob der Pubset große Volumes enthalten darf. Bei der Standardeinstellung \*UNCHANGED bleibt die bisherige Einstellung erhalten. Mit der Angabe \*ALLOWED darf der Pubset große Volumes enthalten. Diese Einstellung ist nicht rückgängig zu machen.

Der Operand LARGE-FILES bestimmt, ob auf diesen großen Volumes auch große Dateien angelegt werden dürfen.

Bei der Standardeinstellung \*UNCHANGED bleibt die bisherige Einstellung erhalten. Mit der Angabe \*ALLOWED dürfen große Dateien angelegt werden. Diese Einstellung ist nicht rückgängig zu machen.

Die Änderung der LARGE\_OBJECTS-Attribute wird erst nach der nächsten Inbetriebnahme des Pubsets (mit dem Kommando IMPORT-PUBSET) wirksam.

## Ausgabe der Pubset-Attribute

Mit dem Kommando SHOW-PUBSET-ATTRIBUTES werden Pubset-Eigenschaften ausgegeben.

*Beispiel der Ausgabe von /SHOW-PUBSET-ATTRIBUTES für den Pubset TST*

```

/sh-pub-attr tst
%
%=====
% PVSID      SYSID      SHAREABILITY  CURRENT    DESIGNATED  BACKUP      ALTERNATE
%                               MASTER      MASTER      MASTER      BACKUP
%-----
%  TST       250        *YES          126        126         124         *BY-OPER
%=====
% DEFAULT-STORAGE-TYPE  LARGE VOLUMES  LARGE FILES  SNAPSET LIMIT
%-----
% *PUBLIC-SPACE        *ALLOWED      *FORBIDDEN   0
%=====
%

```

In der S-Variablen-Ausgabe des Kommandos SHOW-PUBSET-ATTRIBUTES werden diese Festlegungen in folgenden zwei S-Variablen angezeigt:

- var(\*LIST).LARGE-VOL
- var(\*LIST).LARGE-FILE

Inhalt beider S-Variablen: \*NOT-ALLOW oder \*ALLOW.

*Beispiel der Ausgabe in die S-Variable TESTOUT*

```

/decl-var testout(type=*struct),mult-elem=*list
/exec-cmd cmd=(show-pub-attr work),text-output=*no,struct-output=testout
/sh-var testout
TESTOUT(*LIST).PUBSET = WORK
TESTOUT(*LIST).PUBSET-TYPE = *STANDARD
TESTOUT(*LIST).CONTROL-VOLSET =
TESTOUT(*LIST).SYS-ID = 250
TESTOUT(*LIST).SHARE = *YES
TESTOUT(*LIST).LARGE-VOL = *ALLOW
TESTOUT(*LIST).LARGE-FILE = *ALLOW
TESTOUT(*LIST).CURR-MASTER = *NONE
TESTOUT(*LIST).DESIGNATED-MASTER = *NONE
TESTOUT(*LIST).BACKUP-MASTER = *NONE
TESTOUT(*LIST).ALT-BACKUP = *NONE

```

### 3.4.2 MODIFY-USER-PUBSET-ATTRIBUTES

Da die maximale Größe eines Volume-Set 4 TB beträgt, kann die maximal Gesamtgröße eines SM-Pubsets  $255 * 4$  TB, also gut 1.000 TB annehmen. Ein Benutzer kann auf einem SM-Pubset mehr als 4 TB Speicherplatz belegen. Mit dem Kommando MODIFY-USER-PUBSET-ATTRIBUTES können Speicherplatzkontingente  $> 4$  TB zugelassen werden:

```
/MODIFY-USER-PUBSET-ATTRIBUTES . . . ,
    PERM-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED, . . . ) ,
    TEMP-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED, . . . ) ,
    WORK-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED, . . . ) ,
```

Die Angabe TOTAL-SPACE=\*UNLIMITED bestimmt, dass das angegebene Speicherplatzkontingent die Grenze von 4 TB überschreiten darf. Die Festlegung kann für den permanenten Speicherplatz (Operand PERM-SPACE-LIMIT), für den temporären Speicherplatz (Operand TEMP-SPACE-LIMIT) oder für den Speicherplatz von Arbeitsdateien (Operand WORK-SPACE-LIMIT) getrennt getroffen werden.

Informationen über vergebene Speicherplatzkontingente zeigt das Kommando SHOW-USER-ATTRIBUTES mit INFORMATION=\*PUBSET-ATTRIBUTES bzw. \*PUBSET-SUMMARY an:

- Die Ausgabefelder PERM-SPACE-LIMIT, TEMP-SPACE-LIMIT und WORK-SPACE-LIMIT enthalten \*UNLIMITED, wenn das entsprechende Kontingent die Grenze von 4 TB überschreiten darf.
- Die S-Variablen var(\*LIST).PERM-TSL, var(\*LIST).TEMP-TSL und var(\*LIST).WORK-TSL zeigen diese Festlegung mit dem Wert \*UNLIM an.

### 3.4.3 Nichtprivilegierte Kommandos

Die Änderungen der folgenden nichtprivilegierten Kommandos sind im Kapitel 4 „Anwender und Programmierer“ ab [Seite 39](#) beschrieben.

ADD-FILE-LINK	Beschreibung auf <a href="#">Seite 41</a>
SHOW-FILE-ATTRIBUTES	Beschreibung auf <a href="#">Seite 42</a>
SHOW-FILE-LINK	Beschreibung auf <a href="#">Seite 41</a>
SHOW-MASTER-CATALOG-ENTRY	Beschreibung auf <a href="#">Seite 41</a>

## 3.5 Ablauffähigkeit von Dienstprogrammen in Umgebungen mit großen Objekten

In diesem Abschnitt wird auf einige Dienstprogramme hingewiesen, die im Umfeld der Systembetreuung von Bedeutung sind. Es werden wiederum nur die Anpassungen für große Dateien und Volumes dargestellt.

Eine vollständige Darstellung finden Sie in den Handbüchern „HSMS“ [8], „FDDRL“ [6], „SPACEOPT“ [14], „IMON“ [9] und – für alle anderen genannten Dienstprogramme – im Handbuch „Dienstprogramme“ [2].

Zu SIR siehe [Seite 25](#), zu SMPGEN siehe [Seite 27](#).

### 3.5.1 HSMS/ARCHIVE

HSMS/ARCHIVE unterstützt das Sichern und Restaurieren von Dateien >32GB.

Beim RESTORE eines Sicherungsbestands, der auch große Dateien enthält, gilt, dass große Dateien nur auf Large-Objects-Pubsets mit LARGE-FILES-ALLOWED, restauriert werden können.

In allen anderen Fällen – also beim Versuch des Wiederherstellens auf Pubsets mit anderen Eigenschaften – übergehen HSMS und ARCHIVE die großen Dateien und geben für jede davon eine Meldung ARC0061 aus:

```
ARC0061   DATEI GROESSER ALS 32GB. DATEI NICHT BEARBEITET
```

#### Sicherungsdateien auf der S1-Ebene

Für die Sicherung und Migration sind S1-Sicherungsdateien auf großen Platten und/oder als große Dateien möglich.

#### Differenzsicherungen und partielle Sicherungen

Bei Differenzsicherungen und partiellen Sicherungen kann der Fall eintreten, dass im Sicherungsbestand ältere Versionen als kleine Dateien, die aktuelleren Versionen jedoch als große Dateien vorhanden sind. Wird ein derartiger Bestand in einer Umgebung restauriert, in der große Dateien nicht zulässig sind, so werden große Dateien von ARCHIVE mit der Meldung ARC0061 (siehe [Seite 36](#)) abgewiesen.

### RESTORE mit Konvertierung K → NK

Dateien mit BLKCTRL=PAMKEY werden beim Restaurieren in eine Umgebung mit NK-Platten automatisch mit Hilfe des Subsystems PAMINT in ein passendes NK-Format (DATA oder NO) konvertiert. Im Rahmen dieser Konvertierung kann die Größe der konvertierten Datei 32 GB überschreiten. PAMINT unterstützt große Dateien. Somit kann – ein Large-Objects-Pubset mit LARGE-FILES-ALLOWED vorausgesetzt – auch in diesem Fall die Konvertierung durchgeführt werden.

### 3.5.2 VOLIN

Es ist zu beachten, dass große Volumes nur Bestandteil eines Large-Objects-Pubsets sein können.

Als Privatplatten sind große Volumes nicht zulässig. Ein Versuch, ein großes Volume als Privatplatte zu initialisieren, wird mit der Meldung NVL0146 abgewiesen:

```
NVL0146  PRIVATPLATTEN MIT EINER KAPAZITAET GROESSER GLEICH 32GB SIND NICHT
        ZULAESSIG
```

### 3.5.3 SPCCNTRL

SPCCNTRL kann Informationen über große Dateien und Volumes ausgeben.

#### *Ausnahme*

Benennt ENTRY in der Anweisung DISPLAY CATALOG,ENTRY eine Datei  $\geq 32$  GB und ist der Anwender nicht der Systemverwalter (TSOS), kann die Information nicht geliefert werden. Der Auftrag wird mit der Meldung SPC0030 abgewiesen:

```
SPC0030  FEHLER IM MAKRO (&00), RETURN CODE = (&01)
```

## 3.6 Fehlerquellen und Konflikte

### 3.6.1 Einschränkungen für Systemdateien

#### SYSEAM-Dateien

Die Dateigröße von SYSEAM-Dateien muss immer  $< 32$  GB sein.

Die einzige Möglichkeit, dass eine SYSEAM-Datei die kritische Grenze von 32 GB überschreiten kann, ist deren explizites Einrichten durch den Systembetreuer mit dem Makro FILE bzw. dem Kommando CREATE-FILE und entsprechender Dateigröße.

Der erste EAM-Zugriff auf diese Datei wird jedoch abgewiesen. Die Datei muss vom Systembetreuer gelöscht und mit erlaubter Größe neu angelegt werden (siehe auch [Seite 29](#)).

#### Paging-Dateien

Die Paging-Datei darf keine Datei  $\geq 32$  GB sein.

### 3.6.2 Einschränkungen für große Volumes

Große Volumes können nur in Large-Objects-Pubsets aufgenommen werden.

Als Privatplatten sind große Volumes nicht zulässig. Der Versuch, ein großes Volume als Privatplatte zu initialisieren, wird mit der folgenden Fehlermeldung abgewiesen:

```
NVL0146    PRIVATPLATTEN MIT EINER KAPAZITAET GROESSER/GLEICH 32 GB SIND NICHT  
           ZULAESSIG
```

---

## 4 Anwender und Programmierer

In diesem Kapitel werden die Erweiterungen der nicht-privilegierten Kommandoschnittstellen und der Assembler-Makroschnittstellen bzgl. großer Dateien ausführlich beschrieben.

Es folgen Hinweise zu RFA und höheren Programmiersprachen, die von den Erweiterungen um große Objekte betroffen sind.

Den Abschluss des Kapitels bildet die Zusammenfassung von Fehlerquellen und Konflikten.

## 4.1 Benutzerkommandos

In diesem Abschnitt beschreibt die Kommandos und deren Operanden, die von großen Objekten betroffen sind.

### Übersicht

Kommando / Operand	Erläuterung
ADD-FILE-LINK EXCEED-32GB=*BY-PROGRAM/ *FORBIDDEN/*ALLOWED	der Operand legt fest, ob die Dateigröße 32 GB überschreiten darf
SHOW-FILE-LINK	die S-Variable var(*LIST).EXC-32GB; zeigt den Inhalt des Attributs EXCEED-32GB an
SHOW-FILE-ATTRIBUTES	Wenn die Summe der reservierten PAM-Seiten 32 GB überschreitet, erfolgt die Anzeige in Tausend PAM-Seiten. Zwei zusätzliche S-Variablen enthalten die Anzahl in Tausend PAM-Seiten, wenn die ursprüngliche S-Variable 2147483647 PAM-Seiten erreicht hat: var(*LIST).MIGRATE-S1-RESERVED-T und var(*LIST).PUBSET-RESERVED-T
SHOW-MASTER-CATALOG-ENTRY	die zwei S-Variablen var(*LIST).LARGE-VOL und var(*LIST).LARGE-FILE zeigen den Inhalt der Attribute LARGE-VOLUMES und LARGE-FILES an

Tabelle 5: Benutzerkommandos



### 4.1.1 SHOW-MASTER-CATALOG-ENTRY

Das Kommando SHOW-MASTER-CATALOG-ENTRY gibt Auskunft über die Inhalte von MRSCAT-Einträgen.

#### Ausgabe der Pubset-Attribute

Wenn der Pubset ein Large-Objects-Pubset ist, werden in der Ausgabe des Kommandos SHOW-MASTER-CATALOG-ENTRY diese Eigenschaften ausgegeben.

*Beispiel der Ausgabe von /SHOW-MASTER-CATALOG-ENTRY für den Pubset TST*

```
/show-master-catalog-entry tst
%PUBSET TST : SINGLE-FEATURE, PUBRES-UNIT=5140, LOCAL-IMPORTED, NK2-FORMAT
%           LARGE-OBJECTS, EXTRA-LARGE-CATALOG
%           SHARED, MASTER-HOST=OWN-HOST
```

### 4.1.2 ADD-FILE-LINK / SHOW-FILE-LINK

#### Kennzeichnung der Zulässigkeit von großen Dateien

Ohne Eingriff in das Programm kann der Zugriff auf große Dateien mit dem Kommando ADD-FILE-LINK und dem Operanden EXCEED-32GB gesteuert werden.

```
/ADD-FILE-LINK SUPPORT=*DISK(EXCEED-32GB=*BY-PROGRAM/*FORBIDDEN/*ALLOWED,...)
```

Der Operand EXCEED-32GB bestimmt, ob die Bearbeitung von großen Dateien zulässig sein soll (\*ALLOWED) oder nicht (\*FORBIDDEN).

Bei der Standardeinstellung \*BY-PROGRAM werden die Angaben im TU-FCB verwendet. Dieser Operand wird in die TFT (Task File Table) eingetragen und beim Öffnen der Datei ausgewertet.

#### Ausgabe der Datei-Attribute

In der S-Variablen-Ausgabe des Kommandos SHOW-FILE-LINK wird in der S-Variablen `var(*LIST).EXC-32GB` die in der TFT vereinbarte Zulässigkeit von großen Dateien ausgegeben (vereinbart mit dem Kommando ADD-FILE-LINK oder dem Makro FILE).

Inhalt der S-Variablen: " (entspricht BY-PROGRAM) oder \*FORBID oder \*ALLOW.

### 4.1.3 SHOW-FILE-ATTRIBUTES

#### Ausgabe der Dateigröße

Die Ausgabefelder des Kommandos SHOW-FILE-ATTRIBUTES, die die PAM-Seiten-Ausgabe betreffen, erlauben die Ausgabe von Dateigrößen  $\geq 32$  GB. Für die Ausgabe stehen 10 Stellen zur Verfügung, wobei führende Nullen nicht angezeigt werden.

Dies betrifft folgende Ausgabefelder bzw. die folgenden Ausgabefelder in den Summenzeilen:

- Dateigröße in der Kopfzeile (Anzahl der für die Datei reservierten PAM-Seiten)
- HIGH-US-PA bei den Dateimerkmalen (Anzahl der von der Datei belegten PAM-Seiten)
- FRE (Summe der für die angezeigte Ausgabemenge reservierten, aber nicht belegten PAM-Seiten)
- REL (Summe der für die angezeigte Ausgabemenge freigebbaren PAM-Seiten)
- RES (Summe der für die angezeigte Ausgabemenge reservierten PAM-Seiten)  
Bei mehr als 2147483647 reservierten PAM-Seiten erfolgt die Anzeige in Tausend PAM-Seiten (8 Stellen und rechtsbündig das Kennzeichen „T“).
- S-Variable `var(*LIST).MIGRATE-S1-RESERVED` und `var(*LIST).PUBSET-RESERVED`:  
Wenn der Wert von 2147483647 PAM-Seiten erreicht ist, enthält die entsprechende S-Variable `var(*LIST).MIGRATE-S1-RESERVED-T` bzw. `var(*LIST).PUBSET-RESERVED-T` den aktuellen Wert in Tausend PAM-Seiten.

Die Anpassung erfolgte für alle Ausgabeziele (OUTPUT=\*SYSOUT/\*SYSLST/\*PRINTER/<dateiname>).

## 4.2 Assembler-Makros

Dieser Abschnitt wird beschreibt die Assembler-Makro-Schnittstellen, die von großen Objekten betroffen sind.

Die genannten Makros sind in den Handbüchern „Makroaufrufe an den Ablaufteil“ [11] (Makro STAMCE) oder „DVS-Makros“ [3] (alle anderen Makros) beschrieben.

### Übersicht Assembler-Makros

Makro / Operand oder RC	Erläuterung	Seite
Ablaufteil-Makros		
STAMCE SELECT= LARGE_OBJECTS/ LARGE_FILES_ALLOWED	Auswahl und Anzeige der Attribute LARGE_OBJECTS und LARGE_FILES_ALLOWED	45
DVS-Makros		
FSTAT VERSION=0/1  X'0576'	Wird FSTAT mit diesen Versionen aufgerufen, muss die Notwendigkeit einer Umstellung auf VERSION=2/3 geprüft werden.  erweiterter Returncode bzgl. großer Dateien	49
OPEN X'0D9D' X'0D00'	Semantikproblem beachten zusätzlicher Returncode zusätzlicher Returncode	53
FCB LARGE_FILE= *FORBIDDEN/*ALLOWED	Nur für Plattendateien: Mit LARGE_FILE wird angegeben, ob die Datei eine „große“ Datei werden darf, deren Größe 32 GB überschreiten kann.	55
FILE EXC32GB= FORBIDDEN/ALLOWED	Nur für Plattendateien; bestimmt, ob die Dateigröße bei der Datenverarbeitung 32 GB überschreiten darf oder nicht	56
RDTFT	Information über die Dateieigenschaft EXC32GB im Ausgabebereich	57

Tabelle 6: Assembler-Makros (Teil 1 von 2)

Makro / Operand oder RC	Erläuterung	Seite
DVS-Makros für spezielle Zugriffsmethoden		
DIV  LARGE_FILE= *FORBIDDEN/*ALLOWED  OFFSET=anzahl  SPAN=anzahl  X'000C' X'0030'	Operand bei FCT=*OPEN: Mit LARGE_FILE wird angegeben, ob die zu eröffnende Datei eine „große“ Datei werden darf, deren Größe 32 GB überschreiten kann.  Operand bei FCT=*MAP/*SAVE/*RESET: Gibt den ersten Block des im virtuellen Adressraum abzubildenden Dateibereichs an. Der Wert für OFFSET wird begrenzt durch die maximale Größe einer Datei, also: – 8388606 bei LARGE_FILE= *FORBIDDEN bzw. – 1073741823 bei LARGE_FILE=*ALLOWED.  Operand bei FCT=*SAVE/*RESET: Gibt die Länge des Dateibereiches in 4-KB-Blöcken an. Der Wert für SPAN wird begrenzt durch die maximale Größe einer Datei, also: – 8388607 bei LARGE_FILE=*FORBIDDEN bzw. – 1073741824 bei LARGE_FILE=*ALLOWED  Returncode INVALID_LARGE_FILE Returncode LARGE_FILE_NOT_SPECIFIED	58
FPAMSRV  LARGE_FILE= *FORBIDDEN/*ALLOWED  X'0015'	Semantikproblem beachten  Operand bei FCT=*OPEN: Mit LARGE_FILE wird angegeben, ob die zu eröffnende Datei eine „große“ Datei werden darf, deren Größe 32 GB überschreiten kann.  Returncode INVALID_LARGE_FILE	62
FPAMACC  BLOCK=zahl  X'0145'	Gibt den direkten dezimalen numerischen Wert für die Nummer des ersten zu übertragenden logischen Blockes an. Der Wert für BLOCK wird begrenzt durch die maximale Größe einer Datei, also: – 8388606 bei LARGE_FILE=*FORBIDDEN bzw. – 1073741823 bei LARGE_FILE=*ALLOWED (angegeben beim Makro FPAMSRV).  Returncode LARGE_FILE_NOT_SPECIFIED	64

Tabelle 6: Assembler-Makros (Teil 2 von 2)

## 4.2.1 STAMCE

### Ausgabe der Pubset-Attribute

Der Makro STAMCE gibt Auskunft über die Inhalte von MRSCAT-Einträgen und zeigt darüber hinaus die LARGE\_OBJECTS-Attribute an.

*Ausschnitt aus der Dsect von STAMCE*

```
                STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
:
:
:
2 *
2 DMCFDATT    DS      X      attribute
2 DMCFDLOB    EQU    X'40'    set: large_objects
2 *                               files/volumes with more than 32 GB
2 DMCFDLFA    EQU    X'20'    set: large_files_allowed
2 DMCFDRAI    EQU    X'10'    set: pubset with RAID volumes
2 DMCFDGSV    EQU    X'08'    set: gs volumes
2 DMCFDDRV    EQU    X'02'    set: high availability by DRV
2 DMCFDKEY    EQU    X'01'    set: key pubset
2 *
:
```

## Beispiel

Die Ausgabe der mit `DMCFDATT` adressierten Attribute wird so aufbereitet, dass in den Ausgabespalten **YES** für „erlaubt/vorhanden“ und **NO** für „nicht erlaubt/nicht vorhanden“ steht. Die `LARGE_OBJECTS`-Attribute werden in den Spalten `LOB` und `LFA` ausgegeben.

```

STAMCE  START
        PRINT  NOGEN
        BALR   10,0
        USING *,10
        USING DSTAM3,6
        USING DSTAM4,7
        STAMCE MF=E,PARAM=STAM3,VERSION=5
        LR    7,1
        L     6,DMCEAREA
        WROUT HEADER,WROUTERR
SHOW    MVC   ATTRIB,=C'NO NO NO NO NO NO '
        MVC   CATID,DMCFSCTD
        MVC   PROCESS,DMCFSBCA
TLOB    TM    DMCFDATT,DMCFDLOB
        BZ    TLFA
        MVC   LOB,YES
TLFA    TM    DMCFDATT,DMCFDLFA
        BZ    TRAI
        MVC   LFA,YES
TRAI    TM    DMCFDATT,DMCFDRAI
        BZ    TGSV
        MVC   RAI,YES
TGSV    TM    DMCFDATT,DMCFDGSV
        BZ    TDRV
        MVC   GSV,YES
TDRV    TM    DMCFDATT,DMCFDDRV
        BZ    TKEY
        MVC   DRV,YES
TKEY    TM    DMCFDATT,DMCFDKEY
        BZ    TEND
        MVC   KEY,YES
TEND    NOP    TEND
        CLI   PROCESS,X'00'
        BNE   WROUT2
        MVC   PROCESS,=CL8' '
WROUT2  WROUT  OUTREC,WROUTERR
        LA   6,DMCF#(6)
        CLI  DMCFSCTD,' '
        BNE  SHOW
WROUTERR TERM

```

```
*** DEFINITIONS
STAM3    STAMCE CATID=' ',MF=L,VERSION=5
HEADER   DC      Y(HEADERE-HEADER)
          DC      CL2' '
          DC      CL1' '
          DC      C'CATID  PROCESSOR LOB LFA RAI GSV DRV KEY '
HEADERE  EQU      *
OUTREC   DC      Y(OUTRECE-OUTREC)
          DC      CL2' '
          DC      CL1' '
CATID    DS      CL4
          DC      CL3' '
PROCESS  DS      CL8
          DC      CL2' '
ATTRIB   DS      0CL24
LOB       DS      CL4
LFA       DS      CL4
RAI       DS      CL4
GSV       DS      CL4
DRV       DS      CL4
KEY       DS      CL4
OUTRECE  EQU      *
YES       DC      C'YES'
          LTORG
          DS      0F
OUT       DS      XL4096
DSTAM3   STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
DSTAM4   STAMCE MF=D,PREFIX=D,XPAND=PL,VERSION=5
          END
```

*Aufbereitete Ausgabe*

CATID	PROCESSOR	LOB	LFA	RAI	GSV	DRV	KEY
AARZ		NO	NO	NO	NO	NO	NO
BAU3	KAROBUBE	NO	NO	NO	NO	NO	NO
BLAU		NO	NO	NO	NO	NO	NO
BXT2		YES	NO	NO	NO	NO	NO
:							
:							
WORK		YES	YES	NO	NO	NO	YES
2ATS	PIKASS	NO	NO	YES	NO	NO	YES
2CVC	KAROBUBE	NO	NO	NO	NO	NO	NO
4ARB	D015ZE08	NO	NO	NO	NO	NO	YES
:							
:							



## 4.2.2 FSTAT

Bei Zugriff des Makros FSTAT auf Pubsets mit großen Volumes, auf denen jedoch keine großen Dateien erlaubt sind, verhalten sich die Schnittstellen unverändert. Solche Zugriffe sind also immer problemfrei.

Probleme können entstehen, wenn der Zugriff auf Pubsets erfolgt, auf denen auch große Dateien erlaubt sind.

Der Makro FSTAT bietet folgende Schnittstellenvarianten an:

<b>(a)</b>	Version=0	(entspricht Version 710, Default)	
<b>(b)</b>	Version=1	(entspricht Version 800)	ab BS2000-Version V8.0
<b>(c)</b>	Version=2		ab OSD-BC-Version V1.0
<b>(d)</b>	Version=3		ab OSD-BC-Version V3.0
<b>(e)</b>	Version=4		ab OSD-BC-Version V9.0
<b>(f)</b>	Version=5		ab OSD/BC-Version V11.0

### Schnittstellenvarianten ohne Umstellungsaufwand

Wenn ausschließlich FSTAT-Aufrufe mit Version 2 oder höher vorkommen, treten keine Inkompatibilitäten auf.

Die Varianten ab (c) geben die entsprechenden Daten (Extent-Liste und Datenfelder für File-Size und Last Page Pointer) als 4-Byte-Felder zurück. Diese Schnittstellen müssen bei Verwendung großer Dateien nicht umgestellt werden und sind von den weiteren Ausführungen nicht betroffen. Es muss jedoch das Semantikproblem im [Abschnitt „OPEN“ auf Seite 54](#) beachtet werden. Jeder Anwender dieser Schnittstelle sollte prüfen, ob dieses Problem auf seine Implementierung zutrifft.

Ebenfalls nicht betroffen sind folgende Varianten:

- FSTAT ...,VERSION=0,<teilqualifizierter Dateiname>  
FSTAT ...,VERSION=0,<Dateigenerationsgruppe mit GEN=YES>
- FSTAT ...,VERSION=1,FNAM

### Schnittstellenvarianten mit Prüfungs-/Umstellungsaufwand

Im Folgenden werden die verbleibenden Varianten von (a) und (b) näher betrachtet:

```
FSTAT ...,VERSION=0/1,SHORT/LONG
```

Diese Varianten liefern als Ausgabe die Kataloginformation im Format BS2000 V10.0. Die Extent-Liste und die Datenfelder für File-Size und Last Page Pointer werden bei der Ausgabe nur mit 3 Bytes dargestellt. Layoutänderungen an diesen Schnittstellen sind aus Kompatibilitätsgründen nicht möglich.

Aufrufe, die große Objekte betreffen, führen zu einem Überlauf der 3-Byte-Datenfelder.

An diesem Punkt werden zwei, von der Treffermenge des FSTAT-Aufrufs abhängige Fälle unterschieden:

a) In der Treffermenge (Menge der selektierten Dateien) existiert keine Datei  $\geq$  32 GB.

FSTAT toleriert in diesem Fall den Überlauf des 3-Byte-Datenfeldes der PHP in der Extent-Liste. Die nicht darstellbaren PHPs erhalten den Wert X'FFFFFF' zugewiesen. Es wird dabei davon ausgegangen, dass eine Auswertung der PHPs an den Schnittstellen nie oder äußerst selten erfolgt. Sollte diese Annahme ausnahmsweise nicht zutreffen, muss auf die Schnittstellenversion 2 oder 3 umgestellt werden.

Damit wird Folgendes erreicht:

- Die Einführung großer Volumes kann kompatibel erfolgen, Anwenderprogramme müssen nicht geändert werden.
- FSTAT-Aufrufe mit vollqualifizierten Pfadnamen werden kompatibel (bis auf den Überlauf der PHP in der Extent-Liste) unterstützt.
- ▶ Es ist keine Umstellung nötig.

b) In der Treffermenge existiert mindestens eine Datei  $\geq$  32 GB (FSTAT wird mit teilqualifiziertem Dateinamen oder Wildcards aufgerufen).

Solche Aufrufe werden mit dem folgenden Returncode zurückgewiesen:

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'01'	X'0576'	Selektion enthält große Dateien.

Betroffen sind folgende Typen von FSTAT-Schnittstellen:

Typ	Bemerkung
I FSTAT ...	per Default wird VERSION=0 gesetzt
II FSTAT ...,VERSION=0	
III FSTAT ...,VERSION=1,SHORT	
IV FSTAT ...,VERSION=1, LONG	
V FSTAT ...,VERSION=1	per Default wird der Operand SHORT gesetzt

- ▶ Diese Aufrufe müssen auf `VERSION ≥ 2` umgestellt werden.

Generell wird empfohlen bei neuen Programmen die neuesten FSTAT Versionen zu nutzen. Dies gilt insbesondere, wenn beim FSTAT Net-Storage-Dateien (ab FSTAT VERSION=4) bzw. SAM-Node-Files (ab FSTAT VERSION=5) berücksichtigt werden sollen.

## Übersicht über FSTAT-Aufrufe

FSTAT <vollqualifizierter Pfadname>,VERSION=0/1

1. nur Zugriff auf kleine Dateien (< 32 GB):  
keine Aktion notwendig <sup>1)</sup>
2. Zugriff auch auf große Dateien (≥ 32 GB):  
Falls der Aufruf über einen der Typen I bis V erfolgt, muss auf `VERSION=2/3` umgestellt werden, außer bei Typ I und II mit Angabe einer Dateigenerationsgruppe und `GEN=YES`.

FSTAT <teilqualifizierter Pfadname oder Wildcards im Pfadnamen>,VERSION=0/1

1. in der Treffermenge liegen nur kleine Dateien (< 32 GB):  
keine Aktion notwendig <sup>1)</sup>



### Achtung!

Prüfen Sie sorgfältig, ob die Voraussetzungen immer erfüllt sind.

Dies ist vermutlich nur bei Pfadnamen möglich, die Wildcards in der Katalogkennung haben und deren restliche Bestandteile vollständig (vollqualifiziert) sind.

2. in der Treffermenge können auch große Dateien liegen (≥ 32 GB):  
Falls der Aufruf über einen der Typen III bis V erfolgt, muss auf `VERSION=2/3` umgestellt werden.

### Resümee

Bei Verwendung der FSTAT-Schnittstelle mit `VERSION < 2` und `FORM=LONG` oder `FORM=SHORT` ist eine Umstellung auf die neueste Schnittstellenversion unter folgenden Randbedingungen erforderlich:

- a) Im betroffenen Programm soll mit dem FSTAT-Aufruf auf große Dateien zugegriffen werden können.

<sup>1</sup> „keine Aktion notwendig“ gilt hier für den Fall, dass in der Treffermenge nur kleine Dateien existieren und der Überlauf des 3-Byte-Datenfeldes der PHP in der Extent-Liste keine Probleme bereitet. Andernfalls muss auf `VERSION>2` umgestellt werden!

- b) Der Pfadname im FSTAT-Aufruf ist teilqualifiziert oder enthält Wildcards und es kann nicht ausgeschlossen werden, dass in der Treffermenge große Dateien enthalten sind. Besonders kritisch sind hier Aufrufe mit Wildcard in der Katalogkennung.

Dabei müssen neben der Schnittstelle gegebenenfalls auch Datenstrukturen im Programm umgestellt werden.

### Steuerung über Systemparameter FST32GB

FST32GB hat nur Einfluss auf folgende FSTAT-Schnittstellen:

- Version=0 (entspricht Version=710) bei Angabe eines vollqualifizierten Dateinamens (jedoch nicht bei Angabe einer Dateigenerationsgruppe mit GEN=YES)
- Version=1 (entspricht Version=800), bei der nicht der Operand FNAM spezifiziert wurde

Der Systembetreuer stellt systemglobal ein, ob das Vorhandensein einer Datei  $\geq 32$  GB in der Menge der selektierten Dateien zur Abweisung des FSTAT-Aufrufs mit dem Returncode X'00000576' führt (FST32GB=0, Standardeinstellung) oder ein Überlauf der 3-Byte-Datenfelder generell toleriert wird (FST32GB=1). Im letzten Fall wird den nicht darstellbaren Datenfeldern der Wert X'FFFFFF' zugewiesen.

#### *Hinweis*

Der Systemparameter FST32GB wird nicht ausgewertet, wenn der FSTAT-Indikator (siehe unten) gesetzt ist.

### Steuerung über FSTAT-Indikator

Ein Verhalten wie bei FST32GB=1, also das Ignorieren des Überlaufs, kann schnittstellenspezifisch bei den FSTAT-Typen I bis V über einen Indikator aktiviert werden.

#### *Hinweis*

Der FSTAT-Indikator besitzt eine höhere Priorität als der Systemparameter FST32GB. Wird der FSTAT-Indikator gesetzt, wird FST32GB nicht ausgewertet.

Der Indikator muss direkt in der Parameterliste gesetzt werden; eine Unterstützung im FSTAT-Makro ist nicht vorhanden.

Beschreibung der Bits in den entsprechenden Dsects:

```

                FSTAT MF=D,PARMOD=31,VERSION=710
IDBFLAG2 DS    X
IDBLOPYE EQU   X'04'
                FLAGS 2
                2-2 S LARGE PUBSET ACCESS=YES

```

```

                FSTAT MF=D,PARMOD=31,VERSION=800
IFLAGO  DC    B'10001100'
ILOPY   EQU   X'04'
                2-2 S LARGE PUBSET ACCESS=YES

```

### 4.2.3 OPEN

Der Makro OPEN und die Zugriffsmethoden sind nur von der Einführung großer Dateien, nicht aber von der Einführung großer Volumes betroffen.

Die Schnittstelle OPEN prüft, ob Dateierweiterungen über 32 GB hinaus und das Erstellen von oder Zugriffe auf Dateien  $\geq 32$  GB zulässig sind.

Hierbei gibt es zwei Aspekte:

- a) Abweisen des Zugriffs auf oder der Erzeugung von großen Dateien für Zugriffsmethoden, die eine Bearbeitung von großen Dateien nicht gestatten.
- b) Kennzeichnen, dass ein Programm Dateien  $\geq 32$  GB erzeugen bzw. öffnen kann.

#### Unverträgliche Schnittstellenvarianten

Schnittstellen, an denen 3-Byte-Blocknummern verwendet werden, sind prinzipiell nicht in der Lage, mit Dateien  $\geq 32$  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 Wiedergewinnungsadresse betroffen.
- 24-Bit-Schnittstelle von ISAM

In allen oben aufgeführten Fällen gilt:

- Der Zugriff auf Dateien  $\geq 32$  GB wird mit dem Returncode X'00000D9D' oder X'00000D00' abgewiesen, abhängig von der Größe des für die Datei allokierten Speicherplatzes (FILE\_SIZE).
- Die Überschreitung einer Dateigröße von 32 GB durch Sekundärallokierung wird unterbunden (LARGE\_FILE).

## 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. Für ausführlichere Informationen dazu siehe im Anhang Abschnitt „[Semantische Inkompatibilitäten](#)“ auf Seite 74.

Über die Ausführung des Makros bzgl. großer Dateien informieren folgende Returncodes:

<b>X'cc'</b>	<b>X'bb'</b>	<b>X'aaaa'</b>	<b>Erläuterung</b>
X'00'	X'00'	X'0D9D'	Fehler beim Eröffnen einer Plattendatei.
X'00'	X'00'	X'0D00'	Systemfehler beim Eröffnen einer Datei.

#### 4.2.4 FCB

Der Makro FCB definiert den Dateisteuerblock, der die zentrale Informationsquelle für die Zugriffsmethoden BTAM, ISAM, SAM und UPAM ist.

##### Kennzeichnung der Zulässigkeit von großen Dateien

Auf Programmebene steuert der Operand LARGE\_FILE die Zulässigkeit von großen Dateien.

Operation	Operanden
FCB	: : [ , LARGE_FILE = { $\frac{*FORBIDDEN}{*ALLOWED}$ } ] :

##### LARGE\_FILE

*nur für Plattendateien (Zugriffsmethoden ISAM, SAM und UPAM):*

Der Operand LARGE\_FILE bestimmt, ob die logische Dateigröße 32 GB überschreiten darf oder nicht.

##### = **\*FORBIDDEN**

Voreinstellung: Die logische Dateigröße darf 32 GB nicht überschreiten.

##### = **\*ALLOWED**

*nur für Dateien mit BLKCTRL ≠ PAMKEY:*

Die logische Dateigröße darf 32 GB überschreiten.

## 4.2.5 FILE

### Allokierungsverhalten

Die Unterstützung von Dateien  $\geq 32$  GB ist nur für Nicht-Pamkey-Dateien möglich und nur auf Pubsets mit `LARGE_OBJECTS=TRUE`, `LARGE_FILES_ALLOWED=TRUE`.

Pamkey-Dateien dürfen daher auf diesen Pubsets aus Kompatibilitätsgründen die bisherigen Allokierungsgrenzen nicht überschreiten. Die Pamkey-Eigenschaft einer Datei ist zum FILE-Zeitpunkt nicht verbindlich, sie wird erst beim OPEN der Datei definitiv festgelegt.

Das Verhalten für Dateien auf allen anderen Pubsets (`LARGE_OBJECTS=FALSE` bzw. `TRUE`, `LARGE_FILES_ALLOWED=FALSE`) ändert sich nicht, d.h. die FILE-Schnittstelle weist eine Allokierung über die 32-GB-Grenze ab.

### Kennzeichnung der Zulässigkeit von großen Dateien

Auf Programmebene steuert der Operand `EXC32GB` die Zulässigkeit von großen Dateien. Der Operand hat keine Auswirkungen auf das Allokierungsverhalten der FILE-Schnittstelle.

Operation	Operanden
FILE	: : ,EXC32GB=FORBIDDEN/ALLOWED :

### EXC32GB

*nur für Plattendateien; nur für Nicht-Pamkey-Dateien:*

Der Operand `EXC32GB` bestimmt, ob die Dateigröße bei der Dateiverarbeitung 32 GB überschreiten darf oder nicht. Der Operand wird in die TFT (Task File Table) eingetragen und erst beim Öffnen der Datei mit `OPEN` ausgewertet.

`EXC32GB` hat während der Verarbeitung keinen Einfluss auf die Speicherplatzzuweisung beim FILE-Aufruf.

#### = FORBIDDEN

Die Dateigröße darf 32 GB nicht überschreiten.

#### = ALLOWED

Die Dateigröße darf 32 GB überschreiten.



## 4.2.6 RDTFT

### Ausgabe der Datei-Attribute

Mit dem Makro RDTFT werden Informationen aus der TFT ausgegeben.

```
RDTFT ...,LINK=...,VERSION=2/3
```

Mit den Operanden VERSION=2/3 und PARMOD=31 wird eine Ausgabeliste erzeugt, die zusätzlich Informationen über die Dateieigenschaft EXC32GB (siehe FILE-Makro, [Seite 56](#)) ausgibt.

*Ausschnitt aus der Dsect von RDTFT*

```

                                RDTFT MF=D,PLIST=OUTPUT,VERSION=3
:
:
:
1 ***** FCB AND DEVICE TYPE INFORMATION
1 IDRFINFO DS      OHL2
:
:
1 IDRIND15 DS      C                INDICATOR 15
1 IDRDESCS EQU    X'80'              7-7 S DESTOC SPECIFIED
1 IDRDESCY EQU    X'40'              6-6 S DESTOC = YES
1 IDRCMSGY EQU    X'20'              5-5 S CLOSMG SPECIFIED
1 IDRCMSGG EQU    X'10'              4-4 S CLOSMG = YES
1 IDRTAPWS EQU    X'08'              3-3 S TAPEWR  SPECIFIED
1 IDRTAPWY EQU    X'04'              2-2 S TAPEWR  = DEVICE-BUFFER
1 IDRX32GS EQU    X'02'              1-1 S EXC32GB SPECIFIED
1 IDRX32GA EQU    X'01'              0-0 S EXC32GB = ALLOWED
:

```

## 4.2.7 DIV

Die Zugriffsmethode DIV verwendet (wie auch FASTPAM) zur Parameterübergabe nicht den TU-FCB, sondern eine eigene Parameterliste.

Diese Parameterliste DIV(l) enthält für die Funktion „Dateieröffnung“ u.a. den Operanden LARGE\_FILE. Dabei verhindert der Standardwert \*FORBIDDEN den unkontrollierten Zugriff auf große Dateien. In der Parameterliste wird diese Funktion durch ein Bitfeld dargestellt, das mit B'0' ( $\hat{=}$  \*FORBIDDEN) vorbelegt ist.

Damit wird für Programme, die in Versionen < OSD-BC V5.0 übersetzt wurden, die Vorbelegung mit dem Standardwert \*FORBIDDEN garantiert.

### Kennzeichnung der Zulässigkeit von großen Dateien

Zur Kennzeichnung der Zulässigkeit von großen Dateien dient der Makro DIV mit der Funktion OPEN. Auf Programmebene steuert der Operand LARGE\_FILE die Zulässigkeit von großen Dateien.

Dieser Operand wird in die TFT (Task File Table) eingetragen und wird bei der Eröffnung der Datei mit OPEN ausgewertet.

Operation	Operanden
DIV	$[ , FCT = \left\{ \begin{array}{l} *OPEN \\ \text{adr} \\ (r) \end{array} \right\}]$ : : $[ , LARGE\_FILE = \left\{ \begin{array}{l} *FORBIDDEN \\ *ALLOWED \\ \text{adr} \\ (r) \end{array} \right\}]$ :

**LARGE\_FILE**

Der Operand LARGE\_FILE bestimmt, ob die Dateigröße bei der Datenverarbeitung 32 GB überschreiten darf oder nicht. Der Operand wird in die TFT (Task File Table) eingetragen und erst beim Öffnen der Datei mit OPEN ausgewertet.

Bei der Form MF=L ist nur die direkte Angabe erlaubt.

**= \*FORBIDDEN**

Voreinstellung: Die Dateigröße darf 32 GB nicht überschreiten.

**= \*ALLOWED**

Die Dateigröße darf 32 GB überschreiten.

**= adr / (r)**

Ist die Adresse eines 1 Byte langen Feldes, das den Wert für LARGE\_FILE enthält, bzw. das Register, welches den Wert enthält.

**Angabe des ersten Blocks des Datenbereichs und der Dateilänge**

Mit dem Operanden OFFSET und der Funktion MAP, SAVE oder RESET wird der erste Block des im virtuellen Adressraum abzubildenden Dateibereichs angegeben. In Abhängigkeit von der maximalen Größe einer Datei wird dieser Wert begrenzt.

Mit dem Operanden SPAN und der Funktion SAVE oder RESET wird die Länge des Dateibereiches in 4-KB-Blöcken angegeben. In Abhängigkeit von der maximalen Größe einer Datei wird dieser Wert begrenzt.

Operation	Operanden
DIV	$[ , FCT = \left\{ \begin{array}{l} *MAP/*SAVE/*RESET \\ \text{adr} \\ (r) \end{array} \right\} ]$ : : $[ , OFFSET = \left\{ \begin{array}{l} \text{number} \\ \text{adr} \\ *equ \\ (r) \end{array} \right\} ] [ , SPAN = \left\{ \begin{array}{l} \text{number} \\ \text{adr} \\ *equ \\ (r) \end{array} \right\} ]$ :

**OFFSET**

OFFSET legt zusammen mit SPAN den Dateibereich fest, für den das Fenster angelegt wird. Der Operand OFFSET kann mit den Funktionen MAP, SAVE oder RESET angegeben werden.

Voreinstellung:           OFFSET = 0

Bei der Form MF=L ist nur die direkte Angabe erlaubt.

**= anzahl**

Gibt den ersten Block des im virtuellen Adressraum abzubildenden Dateibereichs an. Der Wert für OFFSET ist begrenzt durch die maximale Größe einer Datei in 4-KB-Seiten minus 1:

$0 \leq \text{anzahl} \leq 8388606$  bei LARGE\_FILE=\*FORBIDDEN

$0 \leq \text{anzahl} \leq 1073741823$  bei LARGE\_FILE=\*ALLOWED

**= adr**

Ist die symbolische Adresse eines 4 Byte langen Feldes, das den numerischen Wert (binär) für die Angabe des ersten Blocks des im virtuellen Adressraum abzubildenden Dateibereichs enthält.

**= \*equ**

Ist ein Equate, das den numerischen Wert für die Angabe des ersten Blocks des im virtuellen Adressraum abzubildenden Dateibereichs darstellt. Dem Namen des Equates muss das Zeichen „\*“ vorausgehen.

**= (r)**

Ist ein Register, das den numerischen Wert für die Angabe des ersten Blocks des im virtuellen Adressraum abzubildenden Dateibereichs enthält.

**SPAN**

SPAN definiert zusammen mit OFFSET den Dateibereich, auf den sich SAVE bezieht. Der Operand SPAN kann mit den Funktionen SAVE oder RESET angegeben werden.

Voreinstellung:           SPAN = 0

Bei der Form MF=L ist nur die direkte Angabe erlaubt.

**= anzahl**

Gibt die Länge des Dateibereiches in 4-KB-Blöcken an. Der Wert für SPAN ist begrenzt durch die maximale Größe einer Datei in 4-KB-Seiten:

$0 \leq \text{anzahl} \leq 8388607$  bei LARGE\_FILE=\*FORBIDDEN

$0 \leq \text{anzahl} \leq 1073741824$  bei LARGE\_FILE=\*ALLOWED

**= adr**

Ist die symbolische Adresse eines 4 Byte langen Feldes, das die Länge des Dateibereiches in 4-KB-Blöcken (binär) enthält.

**= \*equ**

Ist ein Equate, das die Länge des Dateibereiches in 4-KB-Blöcken (binär) angibt. Dem Namen des Equates muss das Zeichen „\*“ vorausgehen.

**= (r)**

Ist ein Register, das die Länge des Dateibereiches in 4-KB-Blöcken (binär) enthält.

Über die Ausführung des Makros bzgl. großer Dateien informieren zusätzliche Return-codes:

<b>X'cc'</b>	<b>X'bb'</b>	<b>X'aaa'</b>	<b>Erläuterung</b>
X'00'	X'01'	X'000C'	Der Wert für LARGE_FILE (OPEN) ist weder *ALLOWED noch *FORBIDDEN.
X'00'	X'40'	X'0030'	Beim Zugriff auf eine Datei im Modus SHARUPD=YES wurde festgestellt, dass die Dateigröße den Wert von 32 GB übersteigt, beim OPEN für diese Datei wurde aber ein Überschreiten von 32 GB nicht erlaubt.

### 4.2.8 FPAMSRV

Die Zugriffsmethode FASTPAM verwendet (wie auch DIV) zur Parameterübergabe nicht den TU-FCB, sondern eine eigene Parameterliste.

Diese Parameterliste FPAMSRV(l) enthält für die Funktion „Dateieröffnung“ u.a. den Operanden LARGE\_FILE=\*FORBIDDEN/\*ALLOWED.

Der Operand steuert, ob die Bearbeitung von großen Dateien zulässig sein soll.

Dabei verhindert der Standardwert \*FORBIDDEN den unkontrollierten Zugriff auf große Dateien. In der Parameterliste wird diese Funktion durch ein Bitfeld dargestellt, das mit B'0' (≠ \*FORBIDDEN) vorgelegt ist.

Damit wird für Programme, die in Versionen < OSD-BC V5.0 übersetzt wurden, die Voreinstellung mit dem Standardwert \*FORBIDDEN garantiert.

#### Kennzeichnung der Zulässigkeit von großen Dateien

Zur Kennzeichnung der Zulässigkeit von großen Dateien dient der Makro FPAMSRV mit der Funktion OPEN und dem Operanden LARGE\_FILE.

Operation	Operanden
FPAMSRV	$[ , FCT = \left\{ \begin{array}{l} *OPEN \\ \text{adr} \\ (r) \end{array} \right\} ]$ : :  $[ , LARGE\_FILE = \left\{ \begin{array}{l} *FORBIDDEN \\ *ALLOWED \\ \text{adr} \\ (r) \end{array} \right\} ]$ :

#### LARGE\_FILE

Gibt an, ob die zu eröffnende Datei eine „große Datei“ werden darf, deren Dateigröße 32 GB überschreiten kann.

Voreinstellung:           LARGE\_FILE = \*FORBIDDEN

Bei der Form MF=L ist nur die direkte Angabe erlaubt.

**= \*FORBIDDEN**

Voreinstellung: Die Datei darf keine „große Datei“ werden.

**= \*ALLOWED**

Die Datei darf eine „große Datei“ werden.

**= adr / (r)**

Ist die Adresse eines 1 Byte langen Feldes, das den Wert für LARGE\_FILE enthält, bzw. das Register, welches den Wert enthält.

Über die Ausführung des Makros bzgl. großer Dateien informiert ein zusätzlicher Returncode:

<b>X'cc'</b>	<b>X'bb'</b>	<b>X'aaa'</b>	<b>Erläuterung</b>
X'00'	X'01'	X'0015'	Funktion nicht ausgeführt. Ungültige Angabe bei LARGE_FILE

### 4.2.9 FPAMACC

Im Operanden BLOCK des Makros FPAMACC kann der direkte dezimale numerische Wert für die Nummer des ersten zu übertragenden logischen Blockes angegeben werden. In Abhängigkeit von der maximalen Größe einer Datei (angegeben beim Makro FPAMSRV) wird dieser Wert begrenzt.

Operation	Operanden
FPAMACC	: : [ ,BLOCK={ { zahl adr (r) } }] :

#### BLOCK

Bestimmt die Nummer des ersten zu übertragenden logischen FASTPAM-Blockes innerhalb der Datei.

Die Blockgröße wurde beim Makro FPAMSRV, Funktion OPEN, mit dem Operanden BLKSIZE bestimmt. Es sind nur ganzzahlige Werte zugelassen.

Bei der Form MF=L ist nur die direkte Angabe erlaubt.

##### = zahl

Ist die direkte Angabe eines dezimalen numerischen Wertes für die Nummer des ersten zu übertragenden logischen Blockes. Der Wert wird begrenzt durch die maximale Größe einer Datei in 4-KB-Seiten minus 1:

$1 \leq \text{zahl} \leq 8388606$  bei LARGE\_FILE=\*FORBIDDEN (siehe Makro FPAMSRV)

$1 \leq \text{zahl} \leq 1073741823$  bei LARGE\_FILE=\*ALLOWED (siehe Makro FPAMSRV)

##### = adr / (r)

Ist die symbolische Adresse eines 4 Byte langen Feldes, das den numerischen Wert enthält (binär), bzw. das Register, welches diese Adresse enthält.

Über die Ausführung des Makros bzgl. großer Dateien informiert ein zusätzlicher Returncode:

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'40'	X'0145'	Beim Zugriff auf eine Datei im Modus SHARUPD=YES wurde festgestellt, dass die Dateigröße den Wert von 32 GB übersteigt, beim OPEN für diese Datei wurde aber ein Überschreiten von 32 GB nicht erlaubt.



## 4.2.10 Besonderheiten bei SHARUPD=YES

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
  - mit der Angabe `EXCEED-32GB=*ALLOWED` beim Kommando `ADD-FILE-LINK`
- nicht vorbereitete Aufrufer:
  - mit der Angabe `LARGE_FILE=*FORBIDDEN` beim Makro FCB
  - mit der Angabe `EXCEED-32GB=*FORBIDDEN` beim Kommando `ADD-FILE-LINK`

### Zugriffsmethoden UPAM, FASTPAM und DIV

Bei den Zugriffsmethoden UPAM, FASTPAM und DIV 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` bzw. in der TFT das Attribut `EXCEED-32GB=*FORBIDDEN` gesetzt ist, wird die Verarbeitung abgebrochen.

- UPAM liefert in diesem Fall den Returncode

```
X'000009AD' FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT
```

bzw. die entsprechende DMS-Meldung

```
DMS09AD FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT
```

- FASTPAM liefert in diesem Fall den folgenden Returncode in seiner eigenen Parameterliste `FPAMACC(I)`

```
X'00400145' LARGE_FILE_NOT_SPECIFIED
Beim Zugriff auf eine Datei im Modus SHARUPD=YES wurde festgestellt,
dass die Dateigröße den Wert von 32 GB übersteigt, beim OPEN für diese
Datei wurde aber ein Überschreiten von 32 GB nicht erlaubt.
```

- DIV liefert in diesem Fall den folgenden Returncode in seiner eigenen Parameterliste `DIV(I)`

```
X'00400030' LARGE_FILE_NOT_SPECIFIED
Beim Zugriff auf eine Datei im Modus SHARUPD=YES wurde festgestellt,
dass die Dateigröße den Wert von 32 GB übersteigt, beim OPEN für diese
Datei wurde aber ein Überschreiten von 32 GB nicht erlaubt.
```

### Zugriffsmethode NK-ISAM

Bei der Zugriffsmethode NK-ISAM wird bei jedem SVC-Einstieg eine Überprüfung der Größe der betroffenen NK-ISAM-Datei anhand der im File Table Entry verankerten Extent-Liste durchgeführt. Wird dabei eine Dateigröße  $\geq 32$  GB ermittelt und hat der Aufrufer in seinem FCB das Attribut LARGE\_FILE=\*FORBIDDEN bzw. in der TFT das Attribut EXCEED-32GB=\*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      FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT
```

Die Zugriffsmethode **K-ISAM** ist von dieser Problematik nicht betroffen.

## 4.3 Hinweise zu in höheren Programmiersprachen erstellten Programmen

Im Zusammenhang mit höheren Programmiersprachen/Programmiersystemen gibt es zwei unterschiedliche Aspekte der Dateibearbeitung:

- Dateibearbeitung im Rahmen der Programmerstellung durch Compiler
- Dateibearbeitung beim Ablauf eines vom einem Compiler/Programmiersystem erzeugten Objekts.

### Dateibearbeitung durch Compiler

Die Unterstützung von Quellprogrammen  $\geq 32$  GB ist weder notwendig noch wünschenswert – Eingabeobjekte dieser Größe dürften kaum handhabbar sein. Das gleiche gilt für die von Compilern erzeugten Ausgabeobjekte wie Objektmodule, Listings und Diagnoseausgaben.

Soweit die entsprechenden Objekte in Bibliotheken verwaltet werden, darf die Größe der Bibliothek 32 GB überschreiten, nicht aber die Größe der Bibliothekselemente.

### Dateibearbeitung durch Laufzeitsysteme

Für die Verträglichkeit einzelner Laufzeitsysteme mit großen Dateien (entsprechend Produktklasse B) wird auf die Dokumentation des betreffenden Laufzeitsystems verwiesen.

### COBOL

Für COBOL-Programme, die mit COBOL85 übersetzt sind, wird der Versuch, mit einer großen Datei zu arbeiten, mit Filestatus '9x' abgewiesen.

Eine Umgehung dieser Einschränkung durch die explizite Angabe des Kommandos `ADD-FILE-LINK ...,EXCEED-32GB=*ALLOWED` ist unzulässig und führt zu undefiniertem Verhalten, insbesondere zur Auslassung notwendiger Prüfungen.

Für Programme, die mit COBOL2000 übersetzt sind, ist der Zugriff auf Dateien  $\geq 32$  GB über die COBOL-Dateiorganisation `SEQUENTIAL`, `LINE SEQUENTIAL` und `INDEXED` möglich.

Die COBOL-Dateiorganisation `RELATIVE` wird nur bei der Abbildung auf `BS2000-ISAM` mit Dateien  $\geq 32$  GB unterstützt.

Für `BS2000`-Dateien sowie `POSIX`-Dateien der Organisation `SEQUENTIAL` und `RELATIVE` gibt es keine durch das Produkt COBOL verursachte Einschränkung auf Dateien  $< 32$  GB. Dateien der Organisation `INDEXED` sind im `POSIX`-Dateisystem weiterhin in der Größe beschränkt.

## C

Für die korrekte Verarbeitung von BS2000-Dateien  $\geq 32$  GB steht die volle Funktionalität der Schnittstellenfamilie open64 zur Verfügung. Für die Beschreibung dieser Schnittstellen wird auf das zugehörige Handbuch „C-Bibliotheksfunktionen für POSIX-Anwendungen“ [1] verwiesen.

## C++

Die im Lieferumfang von CRTE enthaltene Schnittstelle IOSTREAM bietet keine Unterstützung großer Dateien. Die Verwendung von IOSTREAM für das Öffnen großer Dateien schlägt fehl. Standardmäßig wird außer dem Returncode Null keine weitere Information (insbesondere keine Fehlermeldung) geliefert.

Die Variable „errno“ liefert den Wert 3 (DMS ERROR).

Falls C++-Programme Schnittstellen auf große Dateien benötigen, müssen sie mit dem C API (open64) arbeiten (s.o. Abschnitt „C“).

---

## 5 Hinweise zur Migration

In diesem Kapitel werden Hinweise zur Einführung von großen Dateien gegeben. Die dazu notwendigen Schnittstellen sind in den vorangehenden Kapiteln beschrieben worden. Hier soll aufgezeigt werden, in welchen Schritten die Migration erfolgen könnte, und welche Überlegungen in eine entsprechende Planung einfließen sollten.

### 5.1 Vorüberlegungen

Durch das stete Wachstum von Datenbeständen kann die Größe einzelner Dateien die 32 GB-Grenze überschreiten. Solche Entwicklungen müssen sorgfältig beobachtet und extrapoliert werden, um rechtzeitig Anpassungen vornehmen zu können. Grundsätzlich gibt es die Option, eine Umgebung einzuführen, in der Dateien  $\geq 32$  GB bearbeitet werden können.

Falls eine Einführung großer Dateien erfolgen soll, sind folgende Punkte vorab zu klären:

- Welche Dateien werden bzw. sollen über 32 GB wachsen?
- Welche Programme/Anwendungen sollen in einer „Umgebung mit großen Dateien“ ablaufen?
- Welche Programme/Anwendungen operieren auf diesen Dateien?

Für die so ermittelten Programme/Anwendungen muss geprüft werden, ob sie in Gegenwart großer Dateien ablauffähig sind bzw. große Dateien bearbeiten können. Ggf. sind Anpassungen erforderlich. Darauf wird in den nächsten Abschnitten eingegangen.

## 5.2 Umgebung

Zum Betrieb mit großen Dateien muss durch die Systembetreuung eine passende Umgebung bereit gestellt werden. Es ist sicher zweckmäßig, diese Umgebung in einer ersten Phase für Testzwecke bereitzustellen, z.B. auf einer Testanlage oder auf einem Gastsystem unter VM2000.

### Systemvoraussetzungen

Wie bereits im Kapitel 2 „[Große Objekte im BS2000](#)“ dargestellt, können große Dateien nur in speziellen Pubsets angelegt werden, sog. Large-Objects-Pubsets mit dem Attribut `LARGE_FILES_ALLOWED`.

Mindestens ein derartiger Pubset muss daher bereit gestellt werden. Dies kann durch Generierung mit SIR oder durch nachträgliche Attributierung mit dem Kommando `SET-PUBSET-ATTRIBUTES` erfolgen. Letztere Vorgehensweise bietet den Vorteil, dass existierende Daten übernommen und fortgeschrieben werden können.

### Softwarekonfiguration

Die Softwarekonfiguration der aktuellen OSD/BC-Version unterstützt den Betrieb mit großen Dateien.

## 5.3 Programmumstellung

### 5.3.1 Assemblerprogramme

Assemblerprogramme setzen direkt auf den TU-Schnittstellen von BS2000 OSD/BC auf.

1. Für Programme, die in Gegenwart großer Dateien ablauffähig sein sollen (LARGE-FILES-kompatible Programme), können Anpassungen erforderlich werden.
2. Für Programme, die große Dateien bearbeiten sollen, werden i.d.R. Anpassungsmaßnahmen anfallen.

Im ersten Fall muss geprüft werden, ob und in welcher Weise die Programmschnittstelle FSTAT verwendet wird. Wie im Kapitel 4, Abschnitt „FSTAT“ ab [Seite 49](#) ausführlich dargestellt, beherrschen Schnittstellenvarianten < VERSION=2 nur 3-Byte-Blocknummern. Beim Zugriff auf „große“ Dateien liefern diese Schnittstellen den Returncode x'0576'.

Dieses Verhalten ist dann unerwünscht und fehlerhaft, wenn dieser Returncode „zufällig“ auftritt, weil die Treffermenge eines FSTAT-Aufrufs auch große Dateien enthält, die aber gar nicht bearbeitet werden sollen. Dies kann bei FSTAT-Aufrufen mit teilqualifizierten Dateinamen oder Dateinamen mit Wildcards der Fall sein. Besonders kritisch sind Aufrufe mit Wildcards in der Katalogkennung. In diesem Fall beschränkt sich der Aufruf nicht auf den Dateikatalog eines Pubsets, sondern er erfasst u.U. mehrere oder alle im System importierten Pubsets.

Bei der Umstellung kritischer FSTAT-Aufrufe auf eine Schnittstellenvariante größer oder gleich VERSION=2 ist nicht nur eine Anpassung der Zugriffe auf die Felder des Ausgabebereichs erforderlich. Auch die weitere Verwendung hieraus entnommener Zähler oder Blocknummern ist zu überprüfen und ggf. anzupassen.

Im zweiten Fall ist neben einer evtl. notwendigen Anpassung von FSTAT-Aufrufen auch eine Anpassung im Umfeld der Dateibearbeitung erforderlich. Dazu muss zunächst beim OPEN im FCB (oder im Fall von FASTPAM und DIV in der Parameterliste) das Kennzeichen LARGE\_FILES=\*ALLOWED gesetzt werden. Damit wird angezeigt, dass große Dateien verarbeitet werden sollen. Daneben ist aber auch eine Überprüfung und ggf. Anpassung der Programmlogik erforderlich, um sicherzustellen, dass die Bearbeitung großer Dateien fehlerfrei erfolgt. Wo mit blockorientierten Zugriffsmethoden (UPAM, FASTPAM) oder Blocknummern (Wiedergewinnungsadresse in SAM) gearbeitet wird, ist wie oben sicherzustellen, dass die Blocknummern durchgängig als 4-Byte-Felder behandelt werden. Neben diesen direkten Abhängigkeiten kann jedoch in der Programmlogik auch implizit die Annahme enthalten sein, dass eine Blocknummer bzw. eine Datei nicht größer als  $2^{24}-1$  werden kann. Zur Identifizierung derartiger „semantischer Inkompatibilitäten“ kann kein erschöpfendes Regelwerk angegeben werden. Im Sinn einer Checkliste wird im Anhang eine Liste von Beispielen gegeben (Abschnitt „[Semantische Inkompatibilitäten](#)“, [Seite 74](#)).

Falls keine Anpassungen erforderlich sind oder zu Testzwecken kann die Zulässigkeit großer Dateien auch über die Kommandoschnittstelle ADD-FILE-LINK eingestellt werden (siehe [Seite 41](#)).

### 5.3.2 Programme in höheren Programmiersprachen

Die aktuellen Compiler-Versionen und Laufzeitsysteme unterstützen große Dateien.

Wenn Sie ältere Programme für den Betrieb mit großen Dateien verwenden möchten, dann ist eventuell eine Neuübersetzung mit dem aktuellen Compiler notwendig, siehe folgende Abschnitte.

#### **COBOL**

Eine Neuübersetzung ist notwendig, wenn das Programm zuvor mit COBOL2000 < V1.1A übersetzt worden war.

Für weiterführende Informationen siehe Abschnitt „COBOL“ auf [Seite 67](#).

#### **C, C++**

Das Programm muss angepasst, neu übersetzt und gebunden werden, wenn das Programm zuvor mit einer Programmier-Umgebung CRTE < V2.3C erstellt und gebunden worden war.

Für weiterführende Informationen siehe Abschnitt „C“ auf [Seite 68](#).



---

## 6 Anhang

Im Anhang werden folgende Tabellen und Übersichten angeboten:

- Semantische Inkompatibilitäten ([Seite 74](#))
- Meldungen bzgl. großer Objekte ([Seite 75](#))  
neue oder geänderte Meldungen für die Bearbeitung großer Volumes und Dateien, geordnet nach Meldungsschlüsseln

## 6.1 Semantische Inkompatibilitäten

In Kapitel 5 wurde darauf hingewiesen, dass bei der Anpassung von Assemblercode für Dateien  $\geq 32$  GB neben der Umstellung auf 4-Byte-Blockzähler und -nummern auch zu prüfen ist, ob die Programmlogik implizit von der Annahme Gebrauch macht, dass Dateien nicht größer als 32 GB werden können.

Im Folgenden werden einige Problempunkte beispielhaft aufgeführt:

- Die höchste 3-Byte-Blocknummer 'X'FFFFFF' hat eine spezielle Bedeutung.
- „Blocknummern“  $> \text{X}'00FFFFFF'$  repräsentieren nicht Blöcke, sondern andere Objekte.
- Bei Berechnungen mit Blocknummern oder -zählern  $> \text{X}'00FFFFFF'$  kann es zum Überlauf kommen.
- Die Stellenzahl von Ein- oder Ausgabefeldern reicht nicht zur Darstellung beliebig großer Blocknummern oder -zähler.
- 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.

## 6.2 Meldungen bzgl. großer Objekte

Im Folgenden sind Meldungen aufgeführt, auf die im Zusammenhang mit großen Objekten in diesem Handbuch verwiesen wird. Sie sind alphabetisch nach Meldungsschlüsseln geordnet.

- ARC0061    ARC0061 DATEI GROESSER ALS 32GB. DATEI NICHT BEARBEITET
- ? Dateien mit einer Groesse ueber 32GB koennen mit dieser ARCHIVE Version oder in dieser Umgebung nicht verarbeitet werden.
- ! Das Wiederherstellen dieser Dateien kann nur in folgender Umgebung durchgefuehrt werden:
- >= BS2000 OSD V5.0A
  - >= ARCHIVE V6.0A
  - Bei Wiederherstellung muss das Zielpubset Dateien groesser 32GB erlauben.
- DMS037E    FORMAT VON VOLUME '(&02)' NICHT KONSISTENT ZU PUBSET/VOLUMESSET '(&00)' IMPORT-PUBSET MIT FORMAT-FEHLER '(&01)' ABGEBROCHEN.
- ? (&00): Kennung des betroffenen SF-Pubsets bzw. Volumesets.  
(&01): Format-Fehler
- '01': PAMKEY- und NON-PAMKEY-Volumes vorhanden. Ein SF-Pubset bzw. ein Volumeset darf nur aus Volumes des gleichen PAMKEY-Typs bestehen.
  - '02': Volumes verschiedener minimaler E/A-Uebertragungseinheiten vorhanden. Ein SF-Pubset bzw. Volumeset darf nur Volumes gleicher minimaler E/A-Uebertragungseinheiten enthalten.
  - '03': Volumes verschiedener minimaler Allokierungseinheiten vorhanden. Ein SF-Pubset bzw. Volumeset darf nur Volumes gleicher minimaler Allokierungseinheiten enthalten.
  - '04': Ein Volume ist groesser als 32 GB, aber der Pubset hat nicht das Attribut fuer grosse Volumes.
- (&02): aktuell geprueftes Volume
- ! Volume-Zugehoerigkeit zum SF-Pubset/Volumeset ueberpruefen, einzelne Volumes neu initialisieren und IMPORT wiederholen.

- DMS0501 ANGEFORDERTER KATALOG NICHT VERFUEGBAR ODER PLATZHALTERZEICHEN IN BENUTZERKENNUNG
- ? Moegliche Ursachen:
- Fall 1: Bei privaten Platten ist die CL2-Option BMTNUM=0 gesetzt.
  - Fall 2: Beim Zugriff auf Dateien ueber eine MSCF-Verbindung wurde mit Platzhalterzeichen (Wildcards) in der Benutzerkennung gearbeitet.
  - Fall 3: Der benoetigte Katalog ist nicht importiert.
  - Fall 4: Im Pfadnamen wurde die Kennung eines Volumesets aus einem Pubset angegeben.
  - Fall 5: Der benoetigte Volumeset ist nicht verfuegbar.
- ! Fall 1: Den Katalog vom Systemverwalter verfuegbar machen lassen. Die CL2-Option BMTNUM ungleich '0' setzen.
- Fall 2: Aufruf ohne Platzhalterzeichen wiederholen.
  - Fall 3: Den benoetigten Katalog vom Systemverwalter importieren lassen.
  - Fall 4: Die Kennung des entsprechenden Pubsets angeben.
  - Fall 5: Den benoetigten Volumeset vom Systemverwalter verfuegbar machen lassen.
- DMS0854 DIE GROESSE DER SYSEAM-DATEI (&00) DARF 32 GIGABYTES NICHT UEBERSCHREITEN
- ? Die SYSEAM-Datei (&00) wurde vom Systemverwalter mit einer nicht erlaubten Dateigroesse von ueber 32 Gigabytes angelegt (durch CREATE-FILE).
- ! SYSEAM-Datei (&00) loeschen und mit einer erlaubten Dateigroesse kleiner als 32 Gigabytes neu anlegen.
- DMS09AD FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT
- ? Im FCB bzw. ADD-FILE-LINK wurde LARGE\_FILE=\*FORBIDDEN spezifiziert, die Datei ist aber groesser als 32 Gigabytes.
- ! Programm korrigieren. Tritt der Fehler weiterhin auf, Systemverwalter verstaendigen.

- DMS0A23 FILE SIZE GROESSER 32 GIGABYTES IST NICHT ERLAUBT  
? Im FCB bzw. ADD-FILE-LINK wurde LARGE\_FILE=\*FORBIDDEN  
spezifiziert, die Datei ist aber groesser als 32 Gigabytes.  
! Programm korrigieren. Tritt der Fehler weiterhin auf,  
Systemverwalter verstaendigen.  
zu bearbeiten.
- DMS1383 VOLUME INKONSISTENT. FEHLER-TYP '(&00)'. KOMMANDO ABGEWIESEN  
? (&00): Fehlertyp  
'01': Raid-Eigenschaft des Volume passt nicht zum  
Pubset/Volumeset.  
'02': GS-Eigenschaft des Volume passt nicht zum  
Pubset/Volumeset.  
'03': Returncode "bad volume" vom Allocator.  
'04': Dual-Recording-Eigenschaft des Volume passt nicht  
zum Pubset/Volumeset.  
'05': Zeitstempel des vom Slave reservierten Volumes stimmt  
nicht mit dem Master ueberein.  
'06': Ein Volume ist groesser als 32GB, aber der Pubset hat  
nicht das Attribut fuer grosse Volumes.  
! Abhaengig von der Ursache.
- NVL0146 PRIVATPLATTEN MIT EINER KAPAZITAET GROESSER GLEICH 32GB SIND NICHT  
ZULAESSIG  
? keine weitere Information  
! keine
- SIR0308 PLATTE FUER VSN '(&00)' BESITZT KAPAZITAET > 32GB; PLATTE  
ZURUECKGEWIESEN  
? Bei ACTION=\*INSTALL wurde in der Anweisung //DECALRE-PUBSET der  
Operand LARGE-DISKS-ALLOWED=\*NO angegeben oder bei ACTION=\*EXTEND  
besitzt das Pubset ein entsprechendes Attribut. Die angegebene  
Platte besitzt aber eine Kapazitaet ueber 32 GB und kann deswegen  
nicht in das Pubset aufgenommen werden.  
! Bei ACTION=\*INSTALL in der Anweisung //DECLARE-PUBSET den  
Operanden LARGE-DISKS-ALLOWED=\*YES angeben.  
Bei ACTION=\*EXTEND vor dem SIR Lauf das Attribut des Pubset mit  
dem Kommando /SET-PUBSET-ATTRIBUTES ...,LARGE-VOLUMES=\*ALLOWED  
aendern.

SIR0728 KOPIEREN DER DATEI '(&00)' ABNORMAL BEENDET. DATEI HAT EINE GROESSE  
UEBER 32GB

? keine weitere Information

! keine

SPC0030 FEHLER IM MAKRO (&00), RETURN CODE = (&01)

? keine weitere Information

! keine

---

# Literatur

Die Handbücher finden Sie im Internet unter <http://manuals.ts.fujitsu.com>. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie auch in gedruckter Form bestellen.

- [1] **CRTE (BS2000)**  
**C-Bibliotheksfunktionen für POSIX-Anwendungen**  
Referenzhandbuch
- [2] **BS2000 OSD/BC**  
**Dienstprogramme**  
Benutzerhandbuch
- [3] **BS2000 OSD/BC**  
**DVS-Makros**  
Benutzerhandbuch
- [4] **BS2000 OSD/BC**  
**Einführung in das DVS**  
Benutzerhandbuch
- [5] **BS2000 OSD/BC**  
**Einführung in die Systembetreuung**  
Benutzerhandbuch
- [6] **FDDRL (BS2000)**  
Benutzerhandbuch
- [7] **HIPLEX MSCF (BS2000)**  
**BS2000-Rechner im Verbund**  
Benutzerhandbuch
- [8] **HSMS (BS2000)**  
**Band 1 und 2**  
Benutzerhandbuch

- [9] **IMON (BS2000)**  
**Installationsmonitor**  
Benutzerhandbuch
- [10] **BS2000 OSD/BC**  
**Kommandos**  
Benutzerhandbuch
- [11] **BS2000 OSD/BC**  
**Makroaufrufe an den Ablaufteil**  
Benutzerhandbuch
- [12] **Performance Handbuch**  
Benutzerhandbuch
- [13] **POSIX (BS2000)**  
**Grundlagen für Anwender und Systemverwalter**  
Benutzerhandbuch
- [14] **SPACEOPT (BS2000)**  
**Optimierung und Reorganisation von Platten**  
Benutzerhandbuch
- [15] **BS2000 OSD/BC**  
**Systeminstallation**  
Benutzerhandbuch



---

# Stichwörter

4-Byte-Felder 17

## A

Ablauffähigkeit  
von Dienstprogrammen 36

ADD-FILE-LINK 41

Adressbreite 20

Allokierungsverhalten 56

ARCHIVE 36

Assembler 71

Attribute für große Objekte

für Pubsets 12

für Volumes 15

Ausgabe

Datei-Attribute 41, 57

Dateigröße 42

Pubset-Attribute 34, 41, 45

## B

Benutzerprogramme 22

BLKCTRL=PAMKEY 18

## C

C/C++ 68, 72

COBOL 67, 72

Compiler 67

## D

Dateien größer 32 GB 11, 12, 17

Einschränkungen 18

Grundlagen 17

max. Größe 11, 17

Schnittstellen 19

Dateityp long long 20

Dienstprogramme

ARCHIVE 36

HSMS 36

SIR 13, 25

SMPGEN 27

SPCCNTRL 37

VOLIN 15, 37

DIV (Makro) 58

DIV (Zugriffsmethode) 65

## E

Einschränkungen

bei Dateien 18

bei Systemdateien 38

bei Volumes 15, 38

Extent-Liste 18

## F

FASTPAM (Zugriffsmethode) 65

FCB (Makro) 55

FILE (Makro) 56

FPAMACC (Makro) 64

FPAMSRV (Makro) 62

FST32GB (Systemparameter) 31, 52

FSTAT (Makro) 49

FSTAT-Indikator 52

## G

Große Dateien 20

große Dateien siehe Dateien größer 32 GB

große Objekte 11

Große POSIX-Dateien 21

Große POSIX-Dateisysteme 20

große Volumes siehe Volumes größer 32 GB

### H

HSMS 36

### I

Indikator (FSTAT) 52

Inkompatibilität, semantische (OPEN) 54

### K

Katalog- und Dateiverwaltung 28

Katalogeintrag-Erweiterung 17, 28

Kennzeichnung der Zulässigkeit

von großen Dateien 41, 55, 56, 58, 62

von großen Objekten in einem Pubset 33

Kommandos

ADD-FILE-LINK 41

SET-PUBSET-ATTRIBUTES 13, 33

SHOW-FILE-ATTRIBUTES 42

SHOW-FILE-LINK 41

SHOW-MASTER-CATALOG-ENTRY 41

SHOW-PUBSET-ATTRIBUTES 34

Kompatibilität

der Pubsets für große Objekte 12

### L

large file aware 21

large file safe 21

LARGE\_FILES\_ALLOWED (Pubset-Attribut) 12

LARGE\_OBJECTS (Pubset-Attribut) 12

LARGE\_VOLUME (Volume-Attribut) 15

LARGE-FILES-Klassifikation 22

Large-Objects-Pubsets siehe Pubsets für große Objekte

Laufzeitsysteme 67

LHP (Logical Half Page) 18

long long (Dateityp) 20

### M

Makros

DIV 58

FCB 55

FILE 56

FPAMACC 64

FPAMSRV 62

FSTAT 49

OPEN 53

RDTFT 57

STAMCE 45

Migration 69

Programmumstellung 71

Softwarekonfiguration 70

Systemvoraussetzungen 70

### N

NK-ISAM (Zugriffsmethode) 66

### O

Objekt, großes 11

OPEN (Makro) 53

### P

Paging-Datei 18, 38

Parameterservice 31

PHP (Physical Half Page) 18

POSIX-Datei 20

POSIX-Dateisystem 20

Privatplatte 15

Programmumstellung

Assembler 71

C/C++ 72

COBOL 72

Pubset-Verwaltung 24

Pubsets für große Objekte

Attribut LARGE\_FILES\_ALLOWED 12

Attribut LARGE\_OBJECTS 12

erstellen 26

im Verbund 27

Kompatibilität 12

max. Größe 24

mit SIR erstellen 25

neue Pubset-Typen 12

Shared-Pubsets 27

Upgrades 13

Pubsets größer 32 GB

Kapazität 11

### R

RDTFT (Makro) 57

Readme-Datei 9

**S**

## Schnittstellen

- bei großen Volumes 15
  - für große Dateien 19
  - für große Pubsets 14
- semantische Inkompatibilität (OPEN) 54
- SET-PUBSET-ATTRIBUTES 13, 33
- SF-Pubsets 27
- Shared-Pubset-Verbund 27
- SHARUPD=YES 65
- SHOW-FILE-ATTRIBUTES 42
- SHOW-FILE-LINK 41
- SHOW-MASTER-CATALOG-ENTRY 41
- SHOW-PUBSET-ATTRIBUTES 34
- SIR 13, 25
- SM-Pubsets 27
- SMPGEN 27
- Softwarekonfiguration
- Migration 70
- SPCCNTRL 37
- STAMCE (Makro) 45
- SVL (Standard-Volume-Label) 15
- SYSEAM-Dateien 18, 29, 38
- Systemdateien
- Paging-Dateien 38
  - SYSEAM-Dateien 38
- Systemparameter FST32GB 31, 52
- Systemvoraussetzungen für Migration 70

**U**

- UPAM (Zugriffsmethode) 65

**V**

- VOLIN 15, 37
- Volume-Sets größer 32 GB
- Kapazität 11
- Volumes größer 32 GB 11, 15
- Attribut LARGE\_VOLUME 15
  - Einschränkungen 15
  - max. Größe 11
  - Privatplatte 15
  - Schnittstelle 15

