OpenVMS
Extended File Specifications の手引き

AA–RFJEA–TE

1999 年 4 月

本書は，Extended File Specifications の概説書です。システム管理者，アプリケーション開発者，一般ユーザに，Extended File Specifications と従来の OpenVMS 環境との相違点，影響について説明します。

改訂/更新情報: 新規マニュアルです。

ソフトウェア・バージョン: OpenVMS Alpha バージョン 7.2
OpenVMS VAX バージョン 7.2

コンパックコンピュータ株式会社

1999 年 4 月

OpenVMS ドキュメント・セットは CD-ROM でも提供しています。

原典　OpenVMS Guide to Extended File Specifications

目次

図

表

まえがき

対象読者

本書は，システム管理者，アプリケーション開発者， Extended File Specifications を OpenVMS 環境で使用する利用者を対象としています。

本書の構成

本書は，次の章から構成されています。

- 第 1 章では， Extended File Specifications の概要とその機能を説明しています。
- 第 2 章では， OpenVMS 管理者が目にする相違点を説明し， ODS-5 volume に利用者がアクセスする手順を示し，記憶メディアへのバックアップとリストア作業への影響について解説します。
- 第 3 章では， ODS-5 volume 環境において OpenVMS 利用者が目にする相違点を説明します。
- 第 4 章では， OpenVMS アプリケーションの HFS へのサポートを評価する方法を説明します。
- 付録 A では， HFS の機能を使用する際に利用者に注意を喚起させるべき点を説明しています。
- 付録 B には， Extended File Specifications をサポートするための OpenVMS プログラミング・インタフェースの変更点についての詳細な技術情報が含まれています。
- 付録 C には， ISO-Latin 1 文字セット用の文字表記テーブルがあります。

関連資料

Extended File Specifications の関連資料については次のドキュメントを参照してください。

- 『Guide to OpenVMS File Applications』
- 『OpenVMS DCL ディクショナリ: A–M』
- 『OpenVMS DCL ディクショナリ: N–Z』
- 『OpenVMS RTL Library (LIB$) Manual』
- 『OpenVMS Record Management Services Reference Manual』

- 『OpenVMS システム管理者マニュアル (上巻)』
- 『OpenVMS システム管理者マニュアル (下巻)』
- 『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル (上巻)』
- 『OpenVMS システム管理ユーティリティ・リファレンス・マニュアル (下巻)』
- 『OpenVMS System Services Reference Manual: A–GETMSG』
- 『OpenVMS System Services Reference Manual: GETQUI–Z』
- 『OpenVMS Utility Routines Manual』
- Advanced Server for OpenVMS Server Admininstrator's Guide

Open Systems Software Group (OSSG) 製品についての追加情報は，次の OpenVMS WWW サイトをご覧ください。

http://www.openvms.digital.com

本書で使用する表記法

本書では，「OpenVMS」は，「DIGITAL OpenVMS」と同意義です。

VMScluster システムは現在では OpenVMS Cluster システムと呼んでいます。とくに表記のない限り， OpenVMS Cluster は， VMScluster と同じ意味です。

本書では， DECwindows および DECwindows Motif はすべて DECwindows Motif for OpenVMS ソフトウェアを意味します。

また，本書では次の表記法も使用しています。

表記法	意味
Ctrl/x	Ctrl/xという表記は， Ctrl キーを押しながら別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
PF1 x	PF1 xという表記は， PF1 に定義されたキーを押してから，別のキーまたはポインティング・デバイス・ボタンを押すことを示します。
Return	例の中で，キー名が四角で囲まれている場合には，キーボード上でそのキーを押すことを示します。テキストの中では，キー名は四角で囲まれていません。 HTML 形式のドキュメントでは，キー名は四角ではなく，括弧で囲まれています。
. . .	例の中の水平方向の反復記号は，次のいずれかを示します。 • 文中のオプションの引数が省略されている。 • 前出の 1 つまたは複数の項目を繰り返すことができる。 • パラメータや値などの情報をさらに入力できる。

表記法	意味
. . .	垂直方向の反復記号は，コードの例やコマンド形式の中の項目が省略されていることを示します。このように項目が省略されるのは，その項目が説明している内容にとって重要ではないからです。
( )	コマンドの形式の説明において，括弧は，複数のオプションを選択した場合に，選択したオプションを括弧で囲まなければならないことを示しています。
[ ]	コマンドの形式の説明において，大括弧で囲まれた要素は任意のオプションです。オプションをすべて選択しても，いずれか 1 つを選択しても，あるいは 1 つも選択しなくても構いません。ただし，OpenVMS ファイル指定のディレクトリ名の構文や，割り当て文の部分文字列指定の構文の中では，大括弧に囲まれた要素は省略できません。
[|]	コマンド形式の説明では，括弧内の要素を分けている垂直棒線はオプションを 1 つまたは複数選択するか，または何も選択しないことを意味します。
{ }	コマンドの形式の説明において，中括弧で囲まれた要素は必須オプションです。いずれか 1 つのオプションを指定しなければなりません。
太字	太字のテキストは，新しい用語，引数，属性，条件を示しています。
italic text	イタリック体のテキストは，重要な情報を示します。また，システム・メッセージ (たとえば内部エラー*number*)，コマンド・ライン (たとえば/PRODUCER=*name*)，コマンド・パラメータ (たとえば*device-name*) などの変数を示す場合にも使用されます。
UPPERCASE TEXT	英大文字のテキストは，コマンド，ルーチン名，ファイル名，ファイル保護コード名，システム特権の短縮形を示します。
Monospace type	モノスペース・タイプの文字は，コード例および会話型の画面表示を示します。 C プログラミング言語では，テキスト中のモノスペース・タイプの文字は，キーワード，別々にコンパイルされた外部関数およびファイルの名前，構文の要約，または例に示される変数または識別子への参照などを示します。
–	コマンド形式の記述の最後，コマンド・ライン，コード・ラインにおいて，ハイフンは，要求に対する引数がその後の行に続くことを示します。
数字	特に明記しない限り，本文中の数字はすべて 10 進数です。 10 進数以外 (2 進数， 8 進数， 16 進数) は，その旨を明記してあります。

1

Extended File Specifications for OpenVMS の概要

Alpha で実行される OpenVMS バージョン 7.2 により，次の 2 つの構成要素から成る Extended File Specifications がインプリメントされます。

- 以前のバージョンの OpenVMS よりも長いファイル名をサポートし，より幅広い範囲の使用可能な文字を持つ，新しいオプションのボリューム構造 ODS-5
- より深いディレクトリ構造のサポート機能

これらの構成要素を利用することによって， OpenVMS Alpha システムは (以前PATHWORKS for OpenVMS と呼ばれていた Advanced Server for OpenVMS 7.2 を使用して)， Windows 95/98 または Windows NT 環境で使用するファイルと似た名前を持つファイルの格納，管理，サービスの提供，およびアクセスを，これまでよりもはるかに柔軟に行うことができます。

この章では， Extended File Specifications の利点，機能，およびサポートの他，Extended File Specifications 環境における OpenVMS の動作の違いの概要を簡単に説明します。

1.1 Extended File Specifications の利点

Extended File Specifications がサポートする深いディレクトリおよび拡張ファイル名には，次の利点があります。

- Advanced Server for OpenVMS 7.2 (以前の PATHWORKS for OpenVMS) のユーザは，より長いファイル名を格納し，ファイル名の大文字小文字の区別を継承し，より深いディレクトリ構造を使用することができる。このため， OpenVMS ファイル・サーバの使用状況が， Windows 95/98 および Windows NT ユーザにとってより透過的になる。
- OpenVMS システム管理者は， Windows 95/98 および Windows NT ユーザが指定した名前を使用して， OpenVMS システム上のファイル名を表示することができる。
- アプリケーション開発者は，深いディレクトリをサポートしている他の環境からアプリケーションを移植する場合に， OpenVMS 上でも同等の構造を使用できる。

- 長いファイル名への対応と Unicode のサポートにより， OpenVMS バージョン 7.2 は， Windows NT クライアントに対するDCOM サーバとして機能することができる。また， ODS-5 は， DCOM 開発者にとって OpenVMS 環境と Windows NT 環境がより同質になるような機能を提供する。
- OpenVMS 上の JAVA アプリケーションは，JAVA オブジェクト命名標準規則に準拠する。
- OpenVMS の一般ユーザは，長いファイル名，新しい文字のサポート，および小文字によるファイル名と大文字小文字が混在したファイル名の両方の処理機能を利用することができる。

これらの利点は，第 1.2 節で説明する機能によるものです。

1.2 Extended File Specifications の機能

Extended File Specifications は， ODS-5 ボリューム構造と深いディレクトリのサポートという主要な 2 つの機能から構成されています。次の項では，これらの機能について説明します。

1.2.1 ODS-5 ボリューム構造

OpenVMS バージョン 7.2 は， On-Disk Structure Level 5 (ODS-5) をインプリメントしています。この構造は，拡張ファイル名を持つファイルを作成し，格納するための基礎となります。 OpenVMS Alpha システム上でボリュームを ODS-5 に変換するかどうかは，選択することができます。

ODS-5 ボリューム構造には，次の機能があります。

- 長いファイル名
- ファイル名の中で使用できる文字の種類の拡大
- ファイル名の中での大文字と小文字の区別の保存

次の項では，これらの機能について説明します。

1.2.1.1 長いファイル名

ODS-5 ボリューム上のファイル名の長さ (バージョン番号を除く) の上限は， 8 ビット文字では 236， 16 ビット文字では 118 です。 255 バイトよりも長い完全なファイル指定は， ODS-5 用に変更されていないアプリケーションで処理するときに， RMS によって自動的に短縮されます。

拡張ファイル名の詳細については，第 3.1.2 項を参照してください。

1.2.1.2 ファイル名の中で使用できる文字の種類の拡大

OpenVMS 上のファイル名には，これまでよりも広範囲の文字を使用できます。ODS-5 は，8 ビットの ISO Latin-1 文字セットと 16 ビットのUnicode (UCS-2) 文字セットをサポートしています。

ISO LATIN-1 および Unicode (UCS-2) 文字セット

ISO Latin-1 Multinational 文字セットは，OpenVMS バージョン 7.2 以前のバージョンで使用していた従来の ASCII 文字セットのスーパーセットです。拡張ファイルの指定では，8 ビットの ISO Latin-1 Multinational 文字セットにあるすべての文字をファイル指定の中で使用することができます。

ただし，**次の文字は使用できません**。

C0 制御コード (0x00 以上， 0x1F 以下)
二重引用符 (")
アスタリスク (*)
バックスラッシュ (\ )
コロン (:)
左および右の山括弧 (< >)
スラッシュ (/)
疑問符 (?)
縦線 (|)

ODS-5 準拠のファイル指定で，スペースなどの特定の特殊文字をあいまいでない形で入力または表示するには，文字の前にサーカンフレックス (^) を付ける必要があります。

ファイル名の中でのこれらの文字セットの使用方法の詳細については，第 3.1.2 項を参照してください。

1.2.1.3 大文字と小文字の区別の保存

以前のバージョンの OpenVMS では， DCL， RMS，およびファイル・システムは，すべてのファイル指定を大文字に変換していました。 ODS-5 は，ファイル指定の大文字と小文字の区別を保存します。次に例を示します。

```
$ CREATE x.Y
Ctrl/Z
$DIRECTORY

Directory  DISK1:[USER1]

x.Y;1

$
```

ここで示されているように，ファイル名の大文字と小文字の区別は保存されています。

大文字と小文字の区別の詳細については，第 3.1.2.6 項を参照してください。

1.2.2 深いディレクトリ構造

ODS-2 および ODS-5 ボリューム構造はどちらも，ディレクトリの深いネスティングをサポートしています。ただし，次の制限があります。

- ディレクトリのレベルは 255 まで
- それぞれのディレクトリの名前は， 8 ビットで 236 文字，または 16 ビットで 118 文字まで

たとえば，ユーザは次のように深くネストされたディレクトリを作成することができます。

```
$ CREATE/DIRECTORY [.a.b.c.d.e.f.g.h.i.j.k.l.m]
```

ユーザは，次のように長い名前を持つディレクトリを， ODS-5 ボリューム上に作成することができます。

```
$ CREATE/DIRECTORY
[.AVeryLongDirectoryNameWhichHasNothingToDoWithAnythingInParticular]
```

255 バイトよりも長い完全なファイル指定は， ODS-5 用に変更されていないアプリケーションで処理するときに， RMS によって自動的に短縮されます。

1.2.2.1 ディレクトリの命名構文

On an ODS-5 ボリュームでは，ディレクトリ名は， ISO Latin-1 文字セットを使用した場合のファイル名と同じ規則に準拠します。ピリオドと特殊文字は，ディレクトリ名の中で使用することができますが，リテラル文字として認識されるためには，サーカンフレックス (^) を前に付けなければならない場合もあります。

第 3.2 節には，深いディレクトリに関する詳しい情報があります。第 3.6.1 項には，長いディレクトリ名の表示に関する情報があります。

1.3 ODS-5 ボリュームを有効にする場合の注意事項

ODS-5 は， Advanced Server for OpenVMS 7.2 (以前の PATHWORKS for OpenVMS) のユーザだけでなく，DCOM および JAVA アプリケーションのユーザにも拡張ファイル共用機能を提供することを主要な目的としています。

ODS-5 ボリュームが有効になると，一部の新しい機能は，特定のアプリケーションまたはレイヤード・プロダクトだけでなく，システム管理の一部の分野にも影響を与える可能性があります。 ODS-5 ボリュームで使用できるファイル名の新しい構文でも， ODS-2 ボリュームでは完全に活用することができません。バージョン 7.2 以前の Alpha システムは ODS-5 ボリュームにアクセスすることができず， OpenVMS バージョン 7.2 VAX システムの持つ ODS-5 機能は限られているため，バージョンが混在する複合アーキテクチャ OpenVMS クラスタで ODS-5 ボリュームを有効にする場所とその方法については，十分注意しなければなりません。

次の項では，ODS-5 ボリュームを有効にすることによってシステム管理，ユーザ，およびアプリケーションが受ける影響の要約を示します。

1.3.1 システム管理に関する注意事項

深いディレクトリと拡張ファイル名への RMS によるアクセスは，OpenVMS Alpha V7.2 システムにマウントされている ODS-5 ボリュームでのみ可能です。ODS-5 ボリュームを有効にするのは，OpenVMS Alpha V7.2 の共通環境クラスタに限ることをお勧めします。

バージョンが混在する複合アーキテクチャ OpenVMS クラスタで ODS-5 を有効にする場合は，システム管理者は次のような特別な手順に従って，複合アーキテクチャ OpenVMS クラスタで生じる固有の制限事項に注意しなければなりません。

- ユーザは，必ず OpenVMS Alpha V7.2 システムから ODS-5 ファイルおよび深いディレクトリにアクセスしなければならない。以前のバージョンではサポートされていない。
- ユーザは，作成した深いディレクトリを OpenVMS Alpha V7.2 システムでのみ表示することができる。
- バージョン 7.2 以前の OpenVMS Alpha システムでは，ODS-5 ボリュームをマウントすることも，そのボリュームにある ODS-2 および ODS-5 ファイル名を読み込むこともできない。

第 1.3.2 項では，バージョンが混在する OpenVMS クラスタまたは複合アーキテクチャによる OpenVMS クラスタでの，ユーザのための ODS-5 サポートの制限事項について，より詳細に説明しています。

ほとんどの通常のアプリケーションでは拡張ファイル名を使用することができますが，すべての拡張ファイル名を使用できるようにするために変更が必要な場合もあります。特別なアプリケーションがディスクへの物理入出力および論理入出力を使用している場合や，アプリケーションが特に ODS-5 ファイル名または ODS-5 ボリュームにアクセスする必要がある場合は，変更が必要になることがあるため，解析が必要です。完全にサポートされている OpenVMS アプリケーションのリストについては，ウェブ・サイトwww.openvms.digital.com/openvms/os/swroll/72.htmlを参照してください。第 1.3.3 項では，ODS-5 が OpenVMS アプリケーションに与える影響について，より詳細に説明しています。

第 2 章には，Extended File Specifications のサポートのレベルを判断するための詳細情報と，ODS-5 ボリュームが有効になっているシステムを管理するためのガイドラインが示されています。

1.3.2 ユーザに関する注意事項

OpenVMS Alpha バージョン 7.2 システムを使用している場合は，OpenVMS Alpha バージョン 7.2 システムにマウントされている ODS-5 ボリューム上で，Extended File Specifications のすべての機能を利用することができます。

バージョンが混在する OpenVMS クラスタ，または複合アーキテクチャによる OpenVMS クラスタを使用している場合は，ODS-5 の機能が制限されることがあります。第 1.3.2.1 項には，バージョンが混在する OpenVMS クラスタでの制限事項のリストが示されています。第 1.3.2.2 項には，複合アーキテクチャによる OpenVMS クラスタでの制限事項のリストが示されています。

1.3.2.1 バージョンの混在に対するサポート

OpenVMS の以前のバージョンを実行しているシステムでは，ODS-5 ボリュームをマウントしたり，拡張ファイル名を正しく処理できないだけでなく，そもそも拡張ファイル名を表示することができません。

この後の項では，バージョンが混在しているクラスタでの，OpenVMS バージョン 7.2 およびそれ以前のバージョンの OpenVMS のサポートについて説明します。

OpenVMS Alpha バージョン 7.2 システム上のユーザ

OpenVMS Alpha バージョン 7.2 システム上のユーザは，バージョン 7.2 以前のファイルとディレクトリに，これまでと同じようにアクセスすることができます。たとえば，次のような操作はすべて可能です。

- ODS-2 ボリューム上に深いディレクトリ構造を作成し，アクセスする。
- 以前のバージョンの OpenVMS で作成された BACKUP セーブセットを読み込む。
- DECnet を使用して，ODS-5 の名前を持つファイルを，以前のバージョンの OpenVMS を実行しているシステム上の ODS-2 の名前を持つファイルにコピーする。

バージョン 7.2 以前のシステム上のユーザ

バージョンが混在しているクラスタでは，いくつかの制限があります。バージョン 7.2 以前の OpenVMS 上のユーザには，次の制限があります。

- ODS-5 ボリューム上のファイルにアクセスできない。ボリュームが CI バスまたは SCSI バスのどちらを経由して物理的に接続されている場合でも，MSCP サーバまたは QIO サーバのどちらに接続されている場合でも，アクセスできない。
- イメージ・セーブセットを正常に作成したりリストアできない。ただし，ODS-5 セーブセットから ODS-2 準拠のファイル名を正しくリストアすることはできる。

1.3.2.2 複合アーキテクチャのサポート

現在の ODS-2 ボリュームおよびファイル管理機能は，VAX システムでも Alpha バージョン 7.2 システムでも同じです。ただし，VAX システム上では，拡張ファイルの命名と解析はサポートされません。

この後の項では，複合アーキテクチャによるクラスタでの OpenVMS VAX システムおよび Alpha システムのサポートについて説明します。

VAX システム上での Extended File Specifications 機能の制限

複合アーキテクチャによる OpenVMS バージョン 7.2 クラスタでは，OpenVMS バージョン 7.2 VAX システム上での Extended File Specifications 機能は，以下の機能に限定されています。

- ODS-5 ボリュームをマウントする機能
- ODS-5 ボリューム上の ODS-2 準拠のファイルを書き込み，管理する機能
- ODS-5 ファイル指定にアクセスすると，疑似名 (\pISO_LATIN\.???または\pUNICODE\.???) が表示

BACKUP の制限

ユーザは VAX システムから，ODS-5 のイメージ・セーブセットを正常に作成したり，リストアしたりできません。ただし，ODS-5 セーブセットから ODS-2 準拠のファイル名を正しくリストアすることはできます。

1.3.3 アプリケーションに関する注意事項

ODS-5 の機能は，ボリューム単位で選択することができます。ODS-5 ボリュームがシステムで有効にされていない場合には，すべての既存のアプリケーションは，これまでと同じように動作します。ODS-5 ボリュームが有効とされている場合には，次の変更点に注意する必要があります。

- OpenVMS のファイル処理機能とコマンド行解析機能は，ODS-5 ボリューム上の拡張ファイル名を使用する一方で，既存のアプリケーションとの互換性を保つことができるように変更されている。既存のほとんどの通常のアプリケーションでは拡張ファイル名を使用することができるが，すべての拡張ファイル名を使用できるようにするために変更が必要な場合もある。
- 特別なアプリケーションがディスクへの物理入出力および論理入出力を使用している場合は，変更が必要になることがあるため，解析が必要です。アプリケーションが特に ODS-5 ファイル名または ODS-5 ボリュームにアクセスする必要がある場合は，変更が必要かどうかを判断するために解析が必要です。

ODS-5 ボリューム上では，ドキュメントが作成されているインタフェースにコーディングされている既存のアプリケーションおよびレイヤード・プロダクトは，ほとんどの DCL コマンド・プロシージャと同じように，変更なしで動作します。

ただし，ドキュメントが作成されていないインタフェースにコーディングされているアプリケーションや，次のような内容が含まれている場合には，ODS-5 ボリューム上でも同じように動作するために，変更が必要なことがあります。

- 次のような情報を含む，ファイル・システムの内部情報

  ディスク上のデータのレイアウト
  ファイル・ヘッダの内容
  ディレクトリ・ファイルの内容

- 特定のディスク構造に合わせたファイル分析機能
- 区切り文字や有効な文字の配置など，ファイルしての構文に関する仮定
- ファイル指定の大文字と小文字の区別に関する仮定。大文字と小文字が混在するファイル指定，および小文字だけによるファイル指定は大文字に変換されないため，文字列の照合処理に影響を与えることがある。
- ファイル指定が RMS とファイル・システム上で同一であるという仮定

注意

OpenVMS VAX または Alpha システム上で実行され，ODS-5 ボリュームにアクセスするすべての未変更の XQP アプリケーションに対しては，ODS-2 準拠でない Unicode または ISO Latin-1 による名前の代わりに，疑似名が返されます。このため，アプリケーションが予期しない動作をする可能性があります。

ODS-5 ディスクを使用し，XQP インタフェースを経由してファイル名を指定または取得するアプリケーションは，拡張名を持つファイルにアクセスできるように変更する必要があります。

OpenVMS アプリケーションのサポート状態の詳細ついては，第 4 章を参照してください。

1.4 OpenVMS アプリケーションで Extended File Specifications を使用する際の推奨事項

システム管理者は，ODS-5 を有効にする前に次の手順を実行しておくことが必要です。

- ODS-5 に関するすべての注意事項を確認する。
- さまざまに異なる OpenVMS アプリケーションに対するサポート・レベルを理解する。
- ODS-5 をサポートしていないか，または ODS-5 名または ODS-5 ボリュームを使用してテストしていないアプリケーションを分離する。

- 付録 A にある，ユーザの期待を整理するためのガイドラインを確認する。

注意

ODS-5 ディスクを有効にするのは，OpenVMS バージョン 7.2 Alpha の共通環境クラスタに限ることをお勧めします。

2

OpenVMS システムでの拡張ファイル命名機能の管理

Extended File Specifications を使用した OpenVMS システムを管理するには，さまざまに異なる OpenVMS アプリケーションによって提供されているサポートについて理解し，新しい環境を有効にする方法や制御する方法，OpenVMS システム管理ユーティリティへの変更点についても，理解しておく必要があります。この章では，次のトピックを扱います。

- Extended File Specifications をサポートする現在の OpenVMS コマンドおよびユーティリティが提供しているサポート・レベル
- OpenVMS Alpha システム上で Extended File Specifications を有効にする方法
- ODS-5 ボリュームへのユーザ・アクセスを制御する方法
- システム管理ユーティリティへの変更点

2.1 Extended File Specifications のサポート・レベル

ODS-5 に対応する OpenVMS ユーティリティおよびコマンドの予想される動作を判断するために，以下に示すサポート・レベルが設定されています。それぞれのサポートレベルは，拡張 (ODS-5 準拠) ファイル指定を処理する際のユーティリティまたはコマンドの共用可能な動作の概略を示しています。

2.1.1 から 2.1.4 までの項では，完全サポートから非サポートまでの，ODS-5 に対するサポート・レベルの定義が示されています。

2.1.1 完全サポート

ODS-5 を完全にサポートする OpenVMS ユーティリティおよびコマンドは，特に拡張ファイル命名機能のすべての特徴を活用できるように変更されています。これらのユーティリティおよびコマンドは，拡張ファイル指定をエラーなしに，また，大文字と小文字の区別を変更せずに受け付け，処理することができます。[1]

さらに，Extended File Specifications を完全にサポートする OpenVMS コマンドおよびユーティリティは，ディレクトリ ID (DID) またはファイル ID (FID) 形式に短縮せずに，元のフォーム[2]で従来課せられていた 255 バイトの制限を超える長いファイ

[1] 新しいファイルの最初のバージョンを作成するとき，新しいファイルの大文字と小文字の区別は，ユーザが指定したものと一致します。既存のファイルのそれ以降のバージョンを作成するときにも，大文字と小文字の区別は元のバージョンと同じまま残されます。

[2] DCL コマンド行で長いファイル指定を入力した場合は，DCL は従来どおり，コマンド行の長さを 255 バイトに制限します。

ル指定を受け付け，また生成することができます。 Extended File Specifications を完全にサポートする OpenVMS コンポーネントのリストについては，第 3.4 節を参照してください。

2.1.2 省略時のサポート

省略時サポート・レベルの OpenVMS ユーティリティおよびコマンドには，Extended File Specifications を活用するための変更がほとんど，またはまったく加えられていません。これらのユーティリティおよびコマンドは，拡張ファイル指定のほとんどの属性 (新しい文字や深いディレクトリ構造など) を正しく処理します。ただし，ファイル名を作成したり表示するときに，大文字と小文字の区別に誤りが生じる可能性があります。

完全サポート・レベルのユーティリティとは異なり，省略時サポート・レベルのユーティリティは， RMS が提供する DID および FID の短縮機能に依存しています。このため，これらのユーティリティには， DID および FID の短縮に関連した次の制約があります。

- FID の短縮が使用されている環境でのマッチング操作は，必ずしも期待どおりに動作しません。たとえば，ワイルドカードを使用してのマッチング操作は，長いファイル名が数値 FID 短縮形式で表現されている可能性があるので，必ずしもすべてのターゲット・ファイル名が操作対象として認識されないことがあります。この制約は， RMS の外部で実行されるマッチング操作に適用されます。
- ワイルドカードや省略時値は， FID では使用できません。たとえば，次のコマンドは無効です。

  ```
  $ DIRECTORY a[1,2,3]*.txt
  $ COPY a[1,2,3].txt *.txt2
  ```

  FID の短縮形は 1 つのファイルの一意の数値表現であるため，これを使って他のファイルを表現したり，マッチングを実行することはできません。
- FID の短縮形を使ってファイルを作成することはできません。

DID 短縮形の詳細については，第 B.2.2.7 項を参照してください。 FID 短縮形の詳細については，第 B.2.2.8 項を参照してください。

2.1.3 拡張ファイル名非サポート

拡張ファイル名非サポート・レベルでは， OpenVMS ユーティリティおよびコマンドは， OSD-5 ボリューム上で機能することができるものの，処理できる対象が従来のファイル指定に限られます。これらのユーティリティおよびコマンドは，拡張ファイル指定を処理するときに正しく動作することが保証されていないため， ODS-5 ボリューム上では慎重に使用する必要があります。

表 2–1 には，拡張ファイル名または ODS-5 構造の処理に制限があるために，Extended File Specifications をサポートしていない OpenVMS ユーティリティおよびコマンドが示されています。

2.1.4 ODS-5 非サポート

ODS-5 非サポート・レベルでは， OpenVMS ユーティリティおよびコマンドは，拡張ファイル名を処理することができません。これらのユーティリティおよびコマンドは，従来のファイル指定を処理するときでも正しく動作することが保証されていないため， ODS-5 ボリューム上では慎重に使用する必要があります。

表 2–1 には，拡張ファイル名または ODS-5 構造の処理に制限があるため， Extended File Specifications をサポートしていない OpenVMS ユーティリティおよびコマンドが示されています。

表 2–1 サポートされていない OpenVMS コンポーネント

コンポーネント	注意事項
ODS-5 非サポート	
ディスク・デフラグメンタ	特定のデフラグメンテーション・ツールが ODS-5 ボリュームをサポートするように更新されたという記述がある場合を除き，サポートされない。[1]
システム・ディスク	ODS-5 ボリュームとして設定したり初期化してはならない。
拡張ファイル名非サポート	
コード・コンパイラ	拡張ファイル名をオブジェクト・ファイル名として使用することはできない。ただし，コード・コンパイラが，拡張ファイル名をサポートするアプリケーションを作成することはできる。
INSTALL 既知イメージ	拡張ファイル名を持つイメージを，既知イメージとしてインストールしてはならない。
LINK	拡張ファイル名を持つイメージを出力することはできない。
MONITOR	拡張ファイル名を信頼の置ける形で処理することができない。
ネットワーク・ファイル (NET*.DAT)	拡張ファイル名を使った名前に変更してはならない。
オブジェクト・モジュール (.OBJ)	拡張ファイル名を使った名前に変更してはならない。
ページ・ファイルとスワップ・ファイル	拡張ファイル名を使用してはならない。
SYSGEN	拡張ファイル名を使ってパラメータ・ファイルを作成してはならない。

[1]DFO は ODS-5 ボリュームをサポートするように変更されていることに注意してください。

(次ページに続く)

表 2–1 (続き) サポートされていない OpenVMS コンポーネント

コンポーネント	注意事項
拡張ファイル名非サポート	
システム・スタートアップ・ファイル	拡張ファイル名を使った名前に変更してはならない。

2.2 OpenVMS Alpha システム上で Extended File Specifications を有効にする方法

2.2.1, 2.2.2, および 2.2.3 の各項では, OpenVMS システム上でを活用する方法について説明します。

注意

バージョン 7.2 以前の OpenVMS Alpha を実行しているシステムでは, Extended File Specifications を使用することができません。これらのシステムでは, ODS-5 ボリュームをマウントすることができず, OpenVMS ファイル・システムの拡張ファイル名を利用することができません。

2.2.1 RMS の省略時の Extended File Specifications 機能の使用

RMS では, 8 レベルを超える深さのディレクトリを使用することができるほか, ODS-2 および ODS-5 ボリュームの両方で, 新しい RMS API 拡張機能を使用することができます。ただし, 拡張ファイル名を作成できるのは, ODS-5 ボリューム上に限られます。第 2.2.2 項には, 新しい ODS-5 ボリュームを作成したり, ODS-2 ボリュームを ODS-5 ボリュームに変換するための手順が示されています。

ODS-5 ボリューム上では, ユーザおよびプログラムは, 拡張ファイル名を使ったファイルを作成することができます。ただし, 省略時の設定では, DCL (および一部のアプリケーション) は, 必ずしもすべての拡張ファイル名を受け付けるとは限らず, コマンド行に入力された小文字によるファイル名を大文字に変更します。 DCL がすべての拡張ファイル名を受け付けるようにするには, 第 3.4.1 項で説明しているように, DCL に対して拡張解析スタイルを有効にしなければなりません。

第 B.2 節には, RMS の Extended File Specifications 機能に関する詳しい情報が示されています。

2.2.2 ODS-5 ボリュームを有効にする方法

ODS-5 ボリュームを OpenVMS Alpha システム上に作成するには，システム管理者は次のいずれかを実行しなければなりません。

- 新しいボリュームを ODS-5 として初期化する。
- 既存のボリュームを ODS-2 から ODS-5 に変換する

ODS-5 ボリュームを作成すると， Advanced Server for OpenVMS 7.2 (以前の PATHWORKS) クライアントで ODS-5 の属性を利用できるようになります。これらの属性は， OpenVMS で表示し，管理することができます。

第 2.2.2.1 項には，新しい OSD-5 ボリュームの初期化の手順が示されています。第 2.2.2.2 項には，既存のボリュームを ODS-5 に変換する手順が示されています。

注意

ボリューム・セットに新しいボリュームを追加する予定がある場合は，新しいボリュームの構造レベルがボリューム・セットの構造レベルと一致している必要があります。これらが一致していないと， Mount **ユーティリティは次のエラー・メッセージを表示します。**

```
Structure level on device ... is inconsistent with volume set
```

2.2.2.1 新しい ODS-5 ボリュームの初期化

INITIALIZE コマンドを次の形式で実行することによって，新しいボリュームを ODS-5 ボリュームとして初期化することができます。ボリュームを初期化すると，そのボリューム内の現在の内容は消去されることに注意してください。

```
$ INITIALIZE /STRUCTURE_LEVEL=5 device-name volume-label
```

次に例を示します。

```
$ INITIALIZE /STRUCTURE_LEVEL=5 DKA300: DISK1
$ MOUNT DKA300: DISK1 /SYSTEM
%MOUNT-I-MOUNTED, DISK1 mounted on _STAR$DKA300:
```

最初のコマンドは， DKA300: デバイスを ODS-5 ボリュームとして初期化し，ボリューム・ラベル DISK1 を割り当てます。 2 番目のコマンドは，この DISK1 ボリュームを公用ボリュームとしてマウントします。

ボリュームが ODS-5 ボリュームとして初期化されたことをチェックするには，次のようなコマンドを実行し，結果を表示します。

```
$ WRITE SYS$OUTPUT F$GETDVI ("DKA300:","ACPTYPE")
F11V5
```

F11V5 は，このボリュームが ODS-5 であることを示しています。

2.2.2.2 既存のボリュームの ODS-5 への変換

既存のボリュームを ODS-5 に変換するには，次の手順を実行します。

1. ボリュームをクラスタ全体からディスマウントします。次に例を示します。

   ```
   $ DISMOUNT /CLUSTER DKA300:
   ```

2. このボリュームをプライベート・ボリュームとしてマウントします。次に例を示します。

   ```
   $ MOUNT DKA300: DISK1
   %MOUNT-I-MOUNTED, DISK1 mounted on _STAR$DKA300:
   ```

   /SYSTEM 修飾子を指定しないことにより，システムはこのボリュームを公用ボリュームとしてではなくプライベート・ボリュームとしてマウントします。

3. 次のようなコマンドを実行し，結果を表示すると，ボリュームが ODS-2 ボリュームであることを確認することができます。

   ```
   $ WRITE SYS$OUTPUT F$GETDVI ("DKA300:","ACPTYPE")
   F11V2
   ```

   F11V2 は，このボリュームが ODS-2 であることを示しています。

4. このボリュームをバックアップしておくことをお勧めします。第 2.2.3 項で説明されているように，ボリュームを ODS-5 に変更すると，バックアップ・ボリュームをリストアする以外の方法では ODS-2 形式に戻すことができなくなります。次に例を示します。

   ```
   $ BACKUP /IMAGE DKA300: SAV.BCK /SAVE_SET
   ```

5. 次の形式でコマンドを使用して，ボリュームの特性を設定します。

   ```
   $ SET VOLUME /STRUCTURE_LEVEL=5  device-name
   ```

   次に例を示します。

   ```
   $ SET VOLUME /STRUCTURE_LEVEL=5 DKA300:
   ```

   注意

   SET VOLUME コマンドを使用してボリュームを ODS-5 から ODS-2 に変更することはできません。ボリュームを ODS-2 に戻す方法については，第 2.2.3 項の手順を参照してください。

   SET VOLUME/STRUCTURE_LEVEL コマンドを実行した後で障害が発生した場合は，手順 5 以降の説明を参照してください。

   SET VOLUME コマンドを実行すると，システムは次のテストを行って，このボリュームを変換できるかどうかをチェックします。

   - デバイスはディスクでなければならず，そのディスク上の構造は ODS-2 または ODS-5 でなければならない。

ボリュームがこれらのテストに失敗すると，システムは次のようなメッセージを表示する。

```
%SET-E-NOTMOD, DKA300: not modified
-SET-E-NOTDISK, device must be a Files-ll format disk
```

```
%SET-E-NOTMOD, DKA300: not modified
-SET-W-INVODSLVL, Invalid on-disk structure level
```

- ディスクはプライベート・ディスクとして所有されていなければならず，所有者のプロセス識別番号 (PID) は， SET VOLUME コマンドを実行したプロセスのものと同一でなければならない。

  ボリュームがこれらのテストに失敗すると，システムは次のようなメッセージを表示する。

```
%SET-E-NOTMOD, DKA300: not modified
-SET-W-NOTPRIVATE, device must be mounted privately
```

- マウント・カウントは，このデバイスがマウントされたのはただ 1 回だけであることを示していなければならない。これにより，このデバイスがクラスタにマウントされることを防ぐことができる。

  ボリュームがこれらのテストに失敗すると，システムは次のようなメッセージを表示する。

```
%SET-E-NOTMOD, DKA300: not modified
-SET-W-NOTONEACCR, device must be mounted with only one accessor
```

6. 次のようなコマンドを実行してプライベート・ボリューム DKA300: をディスマウントし，公用ボリュームとして再マウントします。

```
$ DISMOUNT DKA300:
$ MOUNT /CLUSTER DKA300: DISK1
%MOUNT-I-MOUNTED, DISK1 mounted on _STAR$DKA300:
```

ボリュームが ODS-5 に変換されたことをチェックするには，次のようなコマンドを実行し，結果を表示します。

```
$ WRITE SYS$OUTPUT F$GETDVI ("DKA300:","ACPTYPE")
F11V5
```

F11V5 は，このボリュームが ODS-5 であることを示しています。

障害が起こった場合...

SET VOLUME/STRUCTURE_LEVEL コマンドを入力した後で，しかもコマンドが実行される前に入出力エラーのような障害やシステムのクラッシュが発生した場合は，ボリュームの一部だけが更新されている可能性があります。この場合は，MOUNT コマンドを入力すると， Mount ユーティリティは次のいずれかのエラー・メッセージを表示します。

```
Inconsistent file structure level on device ...
```

```
Structure level on device ... is inconsistent with volume set
```

どちらかの条件があてはまる場合には，/NOSHARE 修飾子を使ってのみ (あるいは，/NOSHARE は省略時の修飾子であるため，修飾子なしで) MOUNT コマンドを実行できます。これにより，システムは同じエラー・メッセージを，今度は警告として表示します。

エラー状態から復旧するには， SET VOLUME/STRUCTURE_LEVEL=5 コマンドを再実行します。次に，ディスクをディスマウントし，再マウントします。他に方法がない場合には，以前に作成したバックアップ・ボリュームをリストアすることもできます。

2.2.3 ODS-5 から ODS-2 への変換

ファイルおよびマージという 2 種類の BACKUP 操作は， ODS-5 ファイル・イメージから ODS-2 ファイル・イメージへの変換をサポートしています (ファイルおよびマージ操作については，『OpenVMS システム管理者マニュアル』のバックアップの章で説明しています。)。

この後の説明の例では，セーブ・セットとの間で変換を実行すると，変換されたファイルについて，“created as”または“copied as”メッセージが表示されることに注意してください。

- イメージ操作中の変換
  - ODS-5 イメージのセーブ・セットを ODS-2 ディスクにリストアする

    この方式は， ODS-5 ディスクのイメージ・バックアップがあり，それを ODS-2 ディスクにリストアする場合に使用できます。

    次の例のコマンド行では， IMAGE.BCK が ODS-5 セーブ・セット，DKA200: が ODS-2 ディスクです。この変換方式を使用するには，出力ディスクを ODS-2 にあらかじめ初期化しておき，コマンド行に/NOINIT 修飾子を含めなければなりません。

```
$ BACKUP/LOG/IMAGE/CONVERT DKA500:[000000]IMAGE.BCK/SAVE -
_$ DKA200:/NOINIT
```

```
%BACKUP-I-ODS5CONV, structure level 5 files will be converted to structure
        level 2 on DKA200:
-BACKUP-I-ODS5LOSS, conversion may result in loss of structure level 5
        file attributes
%BACKUP-S-CREATED, created DKA200:[000000]000000.DIR;1
%BACKUP-S-CREATED, created DKA200:[000000]BACKUP.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]CONTIN.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]CORIMG.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]SECURITY.SYS;1
%BACKUP-S-CREATED, created MDA2:[000000]TEST_FILES.DIR;1
%BACKUP-S-CREATEDAS, created DKA200:[TEST_FILES]SUB^_^{DIR^}.DIR;1 as
        DKA200:[TEST_FILES]SUB$$DIR$.DIR;1
%BACKUP-S-CREATEDAS, created
        DKA200:[TEST_FILES.SUB^_^{DIR^}]SUB^&_~_FILE_~.DAT;1 as
        DKA200:[TEST_FILES.SUB$$DIR$]SUB$_$_FILE_$.DAT;1
%BACKUP-S-CREATEDAS, created
        DKA200:[TEST_FILES]THIS^_IS^_A^_TEST^{_FILE_^}.DAT;1 as
        DKA200:[TEST_FILES]THIS$IS$A$TEST$_FILE_$.DAT;1
%BACKUP-S-CREATED, created DKA200:[000000]VOLSET.SYS;1
```

– ODS-5 ディスクを ODS-2 イメージのセーブ・セットに保存する

この方式は，バージョン 7.2 以前の OpenVMS のバージョンを実行しているシステムで読み込むことのできる ODS-5 ディスクを， ODS-2 ディスクに保存する場合に使用できます。

次の例では， DKA500: が ODS-5 ディスク， IMAGE.BCK が DKA200: 上の ODS-2 セーブ・セットです。

```
$ BACKUP/LOG/CONVERT/IMAGE DKA500: DKA200:[000000]IMAGE.BCK/SAVE

%BACKUP-I-ODS5CONV, structure level 5 files will be converted to
        structure level 2 on DKA200:
-BACKUP-I-ODS5LOSS, conversion may result in loss of structure level 5
        file attributes
%BACKUP-S-COPIED, copied DKA200:[000000]000000.DIR;1
%BACKUP-S-COPIED, copied DKA200:[000000]BACKUP.SYS;1
%BACKUP-S-HEADCOPIED, copied DKA200:[000000]BADBLK.SYS;1 header
%BACKUP-S-HEADCOPIED, copied DKA200:[000000]BADLOG.SYS;1 header
%BACKUP-S-HEADCOPIED, copied DKA200:[000000]BITMAP.SYS;1 header
%BACKUP-S-COPIED, copied DKA200:[000000]CONTIN.SYS;1
%BACKUP-S-COPIED, copied DKA200:[000000]CORIMG.SYS;1
%BACKUP-S-HEADCOPIED, copied DKA200:[000000]INDEXF.SYS;1 header
%BACKUP-S-COPIED, copied DKA200:[000000]SECURITY.SYS;1
%BACKUP-S-COPIED, copied DKA200:[000000]TEST_FILES.DIR;1
%BACKUP-S-COPIEDAS, copied DKA200:[TEST_FILES]Sub^_^{Dir^}.DIR;1 as
        DKA200:[TEST_FILES]SUB$$DIR$.DIR;1
%BACKUP-S-COPIEDAS, copied
        DKA200:[TEST_FILES.Sub^_^{Dir^}]Sub^&_~_File_~.Dat;1 as
        DKA200:[TEST_FILES.SUB$$DIR$]SUB$_$_FILE_$.DAT;1
%BACKUP-S-COPIEDAS, copied
        DKA200:[TEST_FILES]This^_is^_a^_Test^{_File_^}.Dat;1 as
        DKA200:[TEST_FILES]THIS$IS$A$TEST$_FILE_$.DAT;1
%BACKUP-S-COPIED, copied DKA200:[000000]VOLSET.SYS;1
```

– ディスクの内容を ODS-2 ディスクに“コピー”する

この方式は，中間セーブ・セットを作成せずに ODS-5 ディスクから ODS-2 ディスクを作成する場合に使用できます。

この変換方式を使用するには，出力ディスクを ODS-2 にあらかじめ初期化しておき，コマンド行に/NOINIT 修飾子を含めなければなりません。次の例のコマンド行では， DKA500: が ODS-5 ディスク， DKA200: が ODS-2 ディスクです。

```
$ BACKUP/LOG/CONVERT/IMAGE DKA500: DKA200:/NOINIT

%BACKUP-I-ODS5CONV, structure level 5 files will be converted to
 structure level 2 on DKA200:
-BACKUP-I-ODS5LOSS, conversion may result in loss of structure level 5
 file attributes
%BACKUP-S-CREATED, created DKA200:[000000]000000.DIR;1
%BACKUP-S-CREATED, created DKA200:[000000]BACKUP.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]CONTIN.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]CORIMG.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]SECURITY.SYS;1
%BACKUP-S-CREATED, created DKA200:[000000]TEST_FILES.DIR;1
%BACKUP-S-CREATED, created DKA200:[TEST_FILES]SUB$$DIR$.DIR;1
%BACKUP-S-CREATED, created DKA200:[TEST_FILES]THIS$IS$A$TEST$_FILE_$.DAT;1
%BACKUP-S-CREATED, created DKA200:[000000]VOLSET.SYS;1
```

• ファイル操作中の変換

– 個別の ODS-5 ファイルを ODS-2 ディスクに“コピー”する

この変換方式を使用すると， ODS-5 ディスクと ODS-2 ディスクとの間でファイルを変換することができます。たとえば，“コピー”操作の対象となるディレクトリを，ディスク・ツー・ディスクで選択することができます。

次の例では， DKA500 が ODS-5 ディスク， DKA200 が ODS-2 ディスクです。

```
$ BACKUP/LOG/CONVERT DKA500:[*...]*.*;* DKA200:[*...]*.*;*

%BACKUP-I-ODS5CONV, structure level 5 files will be converted to
        structure level 2 on DKA200:
-BACKUP-I-ODS5LOSS, conversion may result in loss of structure level 5
        file attributes
%BACKUP-S-CREDIR, created directory DKA200:[TEST_FILES.SUB$$DIR$]
%BACKUP-S-CREATED, created DKA200:[TEST_FILES]THIS$IS$A$TEST$_FILE_$.DAT;1
```

– 個別の ODS-5 ファイルを ODS-2 セーブ・セットに保存する

この方式は， ODS-5 ディスクのファイルを，バージョン 7.2 以前の OpenVMS のバージョンを実行しているシステムで読み込むことのできるセーブ・セットに保存する場合に使用できます。

次の例では，DKA500 が ODS-5 ディスク，DKA200 が ODS-2 ディスクです。FILES.BCK は ODS-2 セーブ・セットです。

```
$ BACKUP/LOG/CONVERT DKA500:[*...]*.*;* DKA200:FILES.BCK/SAVE

%BACKUP-I-ODS5CONV, structure level 5 files will be converted to
        structure level 2 on DKA200:
-BACKUP-I-ODS5LOSS, conversion may result in loss of structure level
        5 file attributes
%BACKUP-S-COPIED, copied DKA200:[000000]000000.DIR;1
%BACKUP-S-COPIED, copied DKA200:[000000]TEST_FILES.DIR;1
%BACKUP-S-COPIEDAS, copied DKA200:[TEST_FILES]Sub^_^{Dir^}.DIR;1 as
        DKA200:[TEST_FILES]SUB$$DIR$.DIR;1
%BACKUP-S-COPIEDAS, copied
        DKA200:[TEST_FILES.Sub^_^{Dir^}]Sub^&_~_File_~.Dat;1 as
        DKA200:[TEST_FILES.SUB$$DIR$]SUB$_$_FILE_$.DAT;1
%BACKUP-S-COPIEDAS, copied
        DKA200:[TEST_FILES]This^_is^_a^_Test^{_File_^}.Dat;1 as
        DKA200:[TEST_FILES]THIS$IS$A$TEST$_FILE_$.DAT;1
```

既存のディレクトリ内でファイル名を変換できない場合には，BACKUP はそのファイル名を変換し，リンクされていない状態のまま残します。これにより，後で ANALYZE /DISK /REPAIR を使って[SYSLOST]ディレクトリに復旧することができます。[SYSLOST]では，ファイルには ODS-2 準拠の名前がつけられます。さらに BACKUP によって，次のようなメッセージが表示されます。

```
%BACKUP-I-RECOVCNT, 5 files could not be converted into a directory on DKA100
-BACKUP-I-RECOVCMD, use the Analyze/Disk_Structure/Repair command to recover
     files
```

この場合には，ファイルを[SYSLOST]から適切なディレクトリに移動する必要があります。“created as”ログ・メッセージを参照し，ファイルが論理的に配置されるべき場所を確認し，そこにファイルを手動で格納します。

2.3 ODS-5 ボリュームへのアクセスの制御

システム管理者は，次の制約の一方または両方を設定する場合があります。

- VAX 上のユーザが ODS-5 ボリューム上のファイルにアクセスすることを禁止する。
- テストされていないアプリケーションから ODS-5 ディスク上のファイルにアクセスすることを禁止する (特定のユーザが ODS-5 ボリューム上でのこのアクセス制御を無効にできるように，設定することも可能)。

システム管理者は，通常の OpenVMS の個別制御を使用して，これらの制約を設定することができます。詳細については，『OpenVMS Guide to System Security』を参照してください。

2.3.1 および 2.3.2 の項では， ODS-5 ボリュームへのアクセスの制約の例を示しています。

2.3.1 VAX ユーザによる ODS-5 ボリュームへのアクセスの禁止

次の操作を実行することによって，ユーザが VAX ノードから ODS-5 ボリュームにアクセスすることを禁止することができます。

1. OpenVMS VAX ノード上で処理を実行しているユーザを識別する識別子 (VAX_NODE など) を定義します。次に例を示します。

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF> ADD /IDENTIFIER VAX_NODE
%UAF-I-RDBADDMSG, identifier VAX_NODE value %X80010037 added to
        rights database
```

2. 各 VAX ノード上で，システム・ライト・リストに VAX_NODE を追加します。次に例を示します。

```
$SET RIGHTS_LIST /ENABLE /SYSTEM VAX_NODE
```

   コマンドの/ENABLE 識別子により， VAX_NODE が，システム・ライト・リストに追加されます。

   さらに，このコマンドを SYSTARTUP_VMS.COM コマンド・プロシージャにも追加します。

3. VAX ノード上のユーザが ODS-5 ボリュームにアクセスできないようにするには，VAX_NODE 識別子の保持者のアクセスを拒否するアクセス制御エントリ (ACE) を設定します。次に例を示します。

```
$ SET SECURITY /CLASS=VOLUME ODS5_DISK -
_$ /ACL=(ID=VAX_NODE,ACCESS=NONE)
```

2.3.2 テストされていないアプリケーションによる ODS-5 ボリュームへのアクセスの禁止

次の操作を実行することによって，テストされていないアプリケーションが ODS-5 ボリュームにアクセスすることを禁止できます。

1. ODS-5 ボリュームへのアクセスを禁止する対象アプリケーションを識別する識別子 (ODS5_UNSAFE など) を定義します。次に例を示します。

```
UAF> ADD /IDENTIFIER ODS5_UNSAFE /ATTR=SUBSYSTEM
%UAF-I-RDBADDMSG, identifier ODS5_UNSAFE value %X80010039 added to
       rights database
```

2. ODS5_UNSAFE 識別子を持つアプリケーションに，保護サブシステムACE を設定します。次に例を示します。

```
$ SET SECURITY /CLASS=FILE SYS$SYSTEM:APPLICATION.EXE -
_$ /ACL=(SUBSYSTEM,ID=ODS5_UNSAFE)
```

3. 各 ODS-5 ボリュームに対して，識別子の保持者による ODS-5 ボリュームへのアクセスを拒否する ACE を設定します。次に例を示します。

```
$ SET SECURITY /CLASS=VOLUME ODS5_DISK/ -
_$ ACL=(ID=ODS5_UNSAFE,ACCESS=NONE)
```

最後の手順の制約を無効にし，訓練を受けたユーザだけが，テストされていないアプリケーションにアクセスできるように設定することもできます。そのためには，次の手順を実行します。

a. 別の識別子を作成します (ODS5_UNTRAINED など)。

```
UAF> ADD /IDENTIFIER ODS5_UNTRAINED
%UAF-I-RDBADDMSG, identifier ODS5_UNTRAINED value %X80010038 added to
       rights database
```

b. この識別子をすべてのユーザに割り当てます。次に例を示します。

```
UAF> GRANT/IDENTIFIER ODS5_UNTRAINED *
%UAF-I-GRANTMSG, identifier ODS5_UNTRAINED granted to *
```

c. 手順 3 の代わりに，ODS5_UNTRAINED 識別子の保持者によるアクセスを拒否するボリュームに対して，アクセス制御エントリ (ACE) を設定します。次に例を示します。

```
$ SET SECURITY /CLASS=VOLUME ODS5_DISK/ -
_$ ACL=(ID=ODS5_UNSAFE+ODS5_UNTRAINED,ACCESS=NONE)
```

このコマンドにより， ODS5_UNTRAINED ユーザは ODS5_UNSAFE アプリケーションを使用してこのボリュームにアクセスできなくなります。

d. ODS-5 ボリューム上でどのようなアプリケーションでも使用できる対象ユーザについては，この識別子を削除します。次に例を示します。

```
UAF> REVOKE/IDENTIFIER ODS5_UNTRAINED SHEILA_USER
%UAF-I-REVOKEMSG, identifier ODS5_UNTRAINED revoked from SHEILA_USER
```

これらの手順を実行すると，次のような結果になります。

- 訓練を受けていないユーザが，テストされていないアプリケーションを使用した場合は， ODS-2 ボリュームに限ってアクセスできる。
- 訓練を受けたユーザは，どのようなアプリケーションを使用した場合でも，ODS-5 ボリュームにアクセスできる。

2.4 システム管理ユーティリティの変更点

この後の項では，拡張ファイル名をサポートするために OpenVMS システム管理ユーティリティに加えられた変更点について説明します。

2.4.1 Analyze/Disk_Structure ユーティリティ

Analyze/Disk_Structure ユーティリティ (ANALYZE/DISK_STRUCTURE) は，Files–11オン・ディスク構造 (ODS) レベル 1， 2，および 5 のディスク・ボリュームの可読性と有効性をチェックするようになりました。

また，次に示す修飾子が新たに追加されました。

/STATISTICS

この修飾子により，有効性チェックの対象となるボリュームに関する統計情報を生成し，ボリュームごとの統計情報が含まれているファイル STATS.DAT が作成されます。 STATS.DAT に含まれる情報は，次のとおりです。

- ボリューム上の ODS-2 ヘッダおよび ODS-5 ヘッダの数
- ODS-5 ボリューム上の特殊ヘッダの数
- ファイル名の長さの分布情報
- 拡張ヘッダ・チェインの分布情報
- ヘッダ識別子の空きスペースの分布情報
- ヘッダ・マップ領域と ACL 領域の空きスペースの分布情報
- 使用中のヘッダ・スペースの合計と未使用のヘッダ・スペースの合計

2.4.2 Backup ユーティリティ (Alpha システムのみ)

Backup ユーティリティ (BACKUP) によって実装された， Alpha システム上で拡張ファイル名をサポートするための新しい機能は，次のとおりです。

- 新しい BACKUP 修飾子: /CONVERT

  ODS レベル 5 ファイルを ODS レベル 2 ファイルに変換すると，一部の ODS-5 ファイル属性が失われることがあります。

注意

/CONVERT 修飾子は，ODS-5 セーブ・セットの ODS-2 ボリュームへのイメージ・リストアを実行するために使用します。出力ボリュームを ODS-2 として保存するには，/NOINIT 修飾子も使用しなければなりません。

- 深いディレクトリ

  深いディレクトリを処理できるように，アルゴリズムが強化されています。OpenVMS バージョン7.2 以前では， Backup ユーティリティによってサポートされているディレクトリは 32 レベルでした。バージョン 7.2 では， Backup ユーティリティは， RMS が許容するレベルまで (現時点では 255 レベル) のディレクトリをサポートしています。

- 拡張文字セット

  BACKUP は，次の文字セットの文字を使用しているファイル名を処理することができる。

  – DEC Multinational (MCS)

  – ISO Latin-1

  – Unicode (UCS-2)

2.4.3 VAX システム上での ODS-5 ボリュームの物理バックアップ

OpenVMS VAX システム上では， BACKUP は，/PHYSICAL 識別子を指定してボリュームをバックアップした場合に限って， ODS-5 ボリュームをサポートします。BACKUP /PHYSICAL コマンドを実行すると， BACKUP は，ディスクの構造化された内容を無視して，ブロック単位でディスクの物理バックアップを作成します。

Alpha システム上では， BACKUP /IMAGE または BACKUP /PHYSICAL コマンドを使用することができます。

Alpha プロセッサ上での Backup ユーティリティによる拡張ファイル名のサポートの詳細については，『OpenVMS システム管理者マニュアル』を参照してください。

2.4.4 Mount ユーティリティ (Alpha システムのみ)

Mount ユーティリティは， ODS-5 ボリュームを完全にサポートするように変更されました。

3
拡張ファイル名の特徴

Extended File Specifications は， Windows NT で利用できるものと同様の，幅広い種類の文字セット・オプションおよび命名規約を提供します。この章では，Extended File Specifications が一般のユーザに与える影響について説明します。この章では次のトピックを扱います。

- ODS-2 と ODS-5 でのファイルおよびディレクトリ指定の違い
- 拡張ファイル名を使用したファイルの操作
- DCL コマンド・プロシージャでの拡張ファイル名の使用
- DECwindows での ODS-5 ファイル指定の表示

3.1 ファイル指定

ODS-5 ボリュームでは，ファイル指定のための命名スタイルとして，**従来型** (ODS-2 準拠) と**拡張型** (ODS-5 準拠) の 2 つがあります。

第 3.1.1 項では， ODS-2 準拠の命名スタイルの構文について説明します。第 3.1.2 項では， ODS-5 準拠の命名スタイルの構文について説明します。

3.1.1 従来型 (ODS-2) 構文

従来型の (ODS-2) ファイル名構文は，ほとんどの OpenVMS ユーザが使い慣れている構文です。 OpenVMS バージョン 7.1 以前ではこの構文が使用され，次の文字セットおよび命名規約がサポートされています。

ODS-2 文字セット
従来の (ODS-2 準拠) ファイル名には，英数字 (A ～ Z， a ～ z， 0 ～ 9)，ドル記号 ($)，アンダスコア (_)，およびハイフン (-) を使用することができます。

大文字と小文字の区別
Case 大文字と小文字の区別は，従来型の構文では保存されません。ファイル名は，大文字，小文字，またはこれら 2 種類を組み合わせて入力できますが，すべての文字は大文字で格納されます。

標準の区切り文字
従来型の構文では，ファイル・タイプの前にピリオド (.) が付きます。ファイル・バージョンは，セミコロン (;) またはピリオド (.) を使用してファイル・タイプと区切られます (システムがファイル指定を表示するときには，ファイル・バージョン番号の前にセミコロンが表示されます)。ディレクトリは大括弧 ([ ]) または山括弧 (<>) で囲まれます。ディレクトリ・レベルはピリオド (.) で区切られます。

ファイルの長さの制限
従来型のファイル指定は 39.39 形式に従っているため，ファイル名とファイル・タイプを区切るために使用できるのは， 1 つのピリオド (.) だけになります。

3.1.2 拡張型 (ODS-5) 構文

ODS-5 ボリュームが提供する拡張ファイル名構文は，より幅広い文字セット，長いファイル名，およびファイル指定をサポートしています。この構文を使用すると，次に示す文字セットや命名規約を使用する Windows NT スタイルのファイル指定を持つファイルを OpenVMS システムに格納し，アクセスすることができます。

3.1.2.1 ISO Latin-1 文字セット

ISO Latin-1 文字セットは，バージョン 7.2 以前の OpenVMS で使用されていた従来の文字セットのスーパーセットです。 8 ビット ISO Latin-1 文字セットの文字は，**次の文字を除いて**すべて使用することができます。

C0 制御コード (0x00 以上 0x1F 以下)
二重引用符 (")
アスタリスク (*)
バックスラッシュ (\ )
コロン (:)
左山括弧 (<)
右山括弧 (>)
スラッシュ (/)
疑問符 (?)
縦線 (|)

3.1.2.2 特殊文字

一部の ISO Latin-1 文字は，特殊機能文字としてではなく，リテラル文字として解釈されるには，ファイル指定の中でエスケープ文字を前に付ける必要があります。RMS および DCL は，拡張ファイル名に含まれているサーカンフレックス (^) をエスケープ文字として解釈します。次のリストは，エスケープ文字を使用する場合の規則を示しています。

- サーカンフレックス (^) の後にアンダスコア (_) またはスペースを続けると，スペースを表す。

- サーカンフレックス (^) の後に次の文字を続けると，続けた文字がファイル指定の中で持つ特別な意味ではなく，ファイル名の一部として使用されることを意味する。

  . , ; [ ] % ^ &

- ユーザがファイル名の中でリテラル文字としてのピリオド (.) を使用する場合は，サーカンフレックス (^) を使用してもしなくてもよい。ピリオドがファイル・タイプとバージョン番号の間の区切り文字として機能する場合を除き，システムはすべてのピリオドに対してサーカンフレックスを追加する。ディレクトリ名の中で使用するリテラル文字としてのピリオド (.) の場合は，その前に**必ず**サーカンフレックスを付ける必要がある。

- エスケープ文字の後に 16 進数が続く場合には， 2 番目の 16 進数が必要になる。その後の 2 文字が，任意の 1 バイト文字の 16 進数値として解釈される。

  たとえば，^20 はスペースを表す。

- ファイル指定の中でエスケープ文字の後に“U”が続く場合には，その後の 4 つの 16 進数が Unicode として解釈されることを表す。例は^U012F。

  ファイル指定の中で前にサーカンフレックス (^) が付いていないすべての文字は，ISO Latin-1 と想定される。

注意

ファイル名に特殊文字が含まれている場合には，VAX システムからアクセスすることができません。複合アーキテクチャ環境の詳細については，第 3.3 節を参照してください。

3.1.2.3 ピリオド (.) の解釈

拡張ファイル名の中でピリオド (.) をリテラル文字として使用するには，ピリオドがファイル名文字であるかまたは区切り文字であるかを， RMS が判断できなければなりません。

拡張ファイル名の中で使用されているピリオド (.) が 1 つだけの場合には，そのピリオドは区切り文字として解釈されます。以前のバージョンの OpenVMS と同様に， 1 つのピリオドの後に数字が続いた場合にも，この処理が実行されます。

```
$ CREATE Test.1
```

このコマンドによって，次のファイルが作成されます。

```
Test.1;1
```

バージョン番号の判断

1 つのファイル名の中に複数のピリオド (.) がある場合には， RMS は，最後のピリオドの後のすべての文字を調べます。これらの文字がすべて数値であるか，すべて数値でありその前にマイナス記号 (-) が付いている場合には，この数値文字列はバージョン番号であると判断されます。ただし， 6 個以上の数字がある場合には， RMS はこのファイル名を無効なファイル名と判断し，受け付けません。最後のピリオドの後に数字でない文字がある場合には，この最後のピリオドはタイプ区切り文字として解釈されます。

たとえば，次のコマンドを使用したとします。`$ CREATE Test4.3.2.1`

この結果，次のファイルが作成されます。`Test4^.3.2;1`

このとき，.2はファイル・タイプ，1はファイル・バージョンです。

バージョン番号がセミコロン (;) によって明示的に区切られている場合には， 5 文字以下の数値文字でなければならず，前にマイナス記号 (-) を付けることができます。

3.1.2.4 ファイル指定の長さの拡大

ODS-5 ボリュームでは，ファイル名とファイル・タイプの合計は， 8 ビットで 236 文字以内，または 16 ビットで 118 文字以内です。 ODS-5 用に変更されていないプログラムやユーティリティでは，ファイル指定の長さが合計で 255 バイト以内に制限されたり，省略される可能性があります。

```
$ CREATE This.File.Name.Has.A.Lot.Of.Periods.DAT
$ CREATE -
_$ ThisIsAVeryLongFileName^&ItWillKeepGoingForLotsAndLotsOfCharacters.Exceed -
_$ ingThe39^,39presentInPreviousVersionsOfOpenVMS
$ DIRECTORY

Directory TEST$ODS5:[TESTING]

ThisIsAVeryLongFileName^&ItWillKeepGoingForLotsAndLotsOfCharacters.Exceeding
The39^,39presentInPreviousVersionsOfOpenVMS;1
This^.File^.Name^.Has^.A^.Lot^.Of^.Periods.DAT;1

Total of 2 files.
```

第 3.6 節では，完全なファイル指定が 255 バイトの上限を超える場合に， RMS がどのようにファイル指定を省略するかについて説明しています。

3.1.2.5 ワイルドカードの使用

ワイルドカードは，単一文字の場合でも複数文字の場合でも， ODS-5 ファイルでは予想どおりに動作します。単一文字のワイルドカードは，ファイル名またはファイル・タイプの中の特定の 1 文字だけを表しますが，ファイル・バージョン文字列の中で使用することはできません。複数文字のワイルドカードは，ファイル名またはファイル・タイプの中の (0 個も含む) 任意の数の文字を表します。複数文字のワイルドカードは，バージョン文字列の代わりに使用することができます。

3.1.2.5.1 **ワイルドカード文字**　次の文字は，OpenVMS 7.2 ボリューム上ではワイルドカード文字として扱われます。

- アスタリスク (*): 複数文字のワイルドカード
- パーセント記号 (%): 単一文字のワイルドカード
- 疑問符 (?): 単一文字のワイルドカード

パーセント記号 (%) は，既存のアプリケーションとの互換性を保つため，引き続き単一文字のワイルドカードとして使用されます。パーセント記号 (%) の前にサーカンフレックス (^) を付けるとリテラル文字として使用することができます。また，Windows NT ファイル名の中では，リテラル文字として使用されます。このため，RMS は，パーセント記号の他に疑問符 (?) も単一文字のワイルドカードとして認識します。疑問符は，OpenVMS 7.2 以降では，パーセント記号とまったく同様に，ワイルドカード文字として機能します。パーセント記号と疑問符は共に，検索パターンの中の1 **文字だけ**とマッチします。

注意

エスケープ文字 (^. など) またはエスケープ・シーケンス (^EF や^U0101 など) は，ワイルドカードのマッチングで使用するための単一文字と考えられています。

3.1.2.5.2 **ワイルドカードの構文**　DCL では拡張ファイル名の大文字と小文字の区別が保存されますが，ワイルドカードのマッチングでは，大文字と小文字は区別されません。

ワイルドカードを含む検索処理では，引き続き，対象となる指定の同じ部分にある対応する文字だけをマッチします。表 3–1 には，ワイルドカードを使用した検索の例を示しています。

表 3–1　ワイルドカードおよびパターン・マッチングの例

パターン	マッチする例	マッチしない例
A*B;*	AHAB.;1	A.B;1
A.*.B*	A^.DISK.BLOCK;1	A^.C^.B.DAT;1
A?B.TXT;*	A^.B.TXT;5	A^.^.B.TXT;1
*.DAT	Lots^.of^.Periods.dat;1	DAT.;1
Mil?no.dat	Milano.dat;1	Millaano.dat;1
NAPOLI.?.DAT	napoli.q.dat;1	napoli.abc77.dat;1

3.1.2.6 大文字と小文字の区別の保存

ODS-5 ボリュームでは，ファイル名の大文字と小文字の区別は，バージョンが異なっても統一されます。ファイル名の大文字と小文字の区別は，そのファイルが最初に作成されたときの状態が保存されます。大文字と小文字の区別が異なるだけで，同じ名

前のファイル名を複数作成すると，DCL は後でできたファイルを新しいバージョンとして扱い，大文字と小文字の区別を元のファイルと同じ状態に変換します。

たとえば，次のコマンドを使用したとします。

```
$ CREATE CaPri.;1
$ CREATE CAPRI
$ CREATE capri
```

この結果，次のファイルが作成されます。

```
CaPri.;1  CaPri.;2  CaPri.;3
```

以前のバージョンの OpenVMS では，DCL および RMS は，すべてのファイル指定を大文字に変換していました。ODS-5 ボリュームでは，すべてのファイル名の大文字と小文字の区別は，ユーザが作成したときの状態のまま保存されます。

3.2 ディレクトリ指定

この後の項では，ODS-5 ボリューム上で使用できる深いディレクトリ構造および拡張命名構文について説明します。これまで OpenVMS でサポートされていた 8 レベルよりも深いディレクトリを使用することができます。

3.2.1 深いディレクトリ構造

OpenVMS 7.2 は，ディレクトリ指定が全体で 8 ビットまたは 16 ビットで 512 文字以内であれば，255 レベルまでのディレクトリのネスティングをサポートしています。

たとえば，ユーザは ODS-2 または ODS-5 ボリューム上で，次のようなディレクトリを作成することができます。

```
$ CREATE/DIRECTORY [a.b.c.d.e.f.g.h.i.j.k.l.m]
```

ユーザは ODS-5 ボリューム上で，長いファイル名を持つ次のようなディレクトリを作成することができます。

```
$ CREATE/DIRECTORY -
[.AVeryLongDirectoryNameWhichHasNothingToDoWithAnythingInParticular]
```

3.2.2 ディレクトリの命名構文

ODS-5 ボリュームでは，ディレクトリ名は， ISO Latin-1 文字セットを使用した場合のファイル名と同じ規則に準拠します。ピリオドと特殊文字は，ディレクトリ名の中で使用することができますが，リテラル文字として認識されるためには，表 3–2 に示すように，エスケープ文字としてのサーカンフレックス (^) を前に付けなければなりません。

表 3–2 ODS-5 ボリューム上のディレクトリ名

CREATE/DIRECTORY. . .	結果
[Hi^&Bye]	Hi^&Bye.DIR;1
[Lots^.Of^.Periods^.In^.This^.Name]	Lots^.Of^.Periods^.In^.This^.Name.DIR;1

3.2.2.1 ディレクトリ ID およびファイル ID の短縮形

状況によっては，完全なファイル指定に，変更されていないアプリケーションで許可されている 255 バイトより多くの文字が含まれている場合があります。そのようなアプリケーションが必要とするファイル指定の長さが 255 バイトを超えている場合，RMS はディレクトリを DID に短縮し，ファイル名を FID に短縮することによって，より短いファイル指定を生成します。

ファイル指定が長すぎる場合， RMS はまずディレクトリをそのディレクトリ ID に変換することにより，より短いディレクトリを生成しようとします。この短い指定は，DID と呼ばれます。

```
TEST$ODS5:[5953,9,0]Alghero.TXT;1
```

UIC 形式のディレクトリ名との混同を避けるため，この形式のディレクトリ名には， 3 つの数字と 2 つのコンマがなければならないことに注意してください。DIRECTORY コマンドを使用すると，ファイル指定の短いバージョンのほか，完全なバージョンも表示することができます。長いファイル指定を表示する方法については，第 3.6 節を参照してください。 DID の短縮形の詳細については，第 B.2.2.7 項を参照してください。 FID の短縮形の詳細については，第 B.2.2.8 項を参照してください。

3.3 複合環境での作業

OpenVMS Alpha システムと OpenVMS VAX システムの両方が含まれている環境で作業を行う場合には，ユーザが次のことを認識していることが重要です。

どちらのタイプのシステムでユーザが作業しているのか
どちらのタイプのボリュームにユーザの省略時のディレクトリが存在するのか
どちらのタイプのボリュームにユーザが新しいファイルを作成するのか

OpenVMS 7.2 を使用すると， VAX システムから ODS-5 ボリュームをマウントすることができます。ただし， OpenVMS VAX システム上のユーザがアクセスできるのは， ODS-2 準拠のファイル名を持つファイルだけです。

ODS-2 ボリュームと ODS-5 ボリュームの複合環境で作業を行う場合には， ODS-5 ボリューム上でファイルを作成するときの ODS-2 ファイル名の制約に注意してください。ファイル名に特殊文字が含まれている ODS-5 ボリューム上のファイルを ODS-2 ボリューム上にコピーするには， ODS-2 準拠の名前を付ける必要があります。

3.4 ODS-5 ボリュームの DCL サポート

拡張ファイル名を DCL コマンド行で使用する場合には，拡張ファイル指定を受け付けて表示できるように，解析スタイルを EXTENDED に設定する必要があります。省略時の設定は TRADITIONAL です。解析スタイルを設定するには，次のコマンドを入力します。

```
$ SET PROCESS/PARSE_STYLE=EXTENDED
```

注意

DCL レキシカル関数は，ODS-5 ディスク上のファイル名に使用されている Latin-1 文字セットとは異なる DEC で定義している文字セットを使用します。このため，たとえば DCL 関数 F$EDIT を使用してファイル名を大文字に変更した場合などには，予期しない結果が発生する可能性があります。F$EDIT は， F0， F7， FE，および FF の 16 進数値を持つ DEC で定義している文字セットの文字については，大文字に変更しません。

DCL の名前解析スタイルの変更の詳細については，第 3.4.1 項を参照してください。

3.4.1 DCL での Extended File Specifications 解析機能の使用

3.4.1.1， 3.4.1.2，および 3.4.1.3 の各項では， DCL 名前解析スタイルを，コマンド行およびコマンド・プロシージャの中で制御する方法について説明します。

3.4.1.1 拡張ファイル名解析機能の設定

OpenVMS Alpha システムでは，次のコマンドを入力して， DCL が ODS-5 ファイル名をプロセス単位で受け付けるように設定することができます。

```
$ SET PROCESS/PARSE_STYLE=EXTENDED
```

このコマンドは， OpenVMS VAX システムには全く影響を与えないことに注目してください。

このコマンドを入力すると， DCL は，次のようなファイル名を受け付けるようになります。

```
$ CREATE MY^[FILE
```

サーカンフレックス (^) は，次に続く文字 (この例の場合には左大括弧 ([)) が区切り記号ではなく，ファイル名の中のリテラル文字として処理されるように DCL に指示するエスケープ文字として使用されます。

詳細については，『OpenVMS DCL ディクショナリ: N–Z』の SET PROCESS /PARSE_STYLE コマンドに関する説明を参照してください。

3.4.1.2 省略時の解析スタイルの再設定

ファイル名に対する省略時の DCL 解析スタイルは， ODS-2 スタイルのファイル名です。 DCL を省略時の解析スタイルに再設定するには，次のコマンドを入力します。

```
$ SET PROCESS/PARSE_STYLE=TRADITIONAL
```

このコマンドを入力すると， DCL は， ODS-2 のファイル名形式だけを受け付けるようになります。

3.4.1.3 ファイル名解析スタイルの切り替え

特定のファイル名解析スタイルを必要とするコマンド・プロシージャでは，そのコマンド・プロシージャ内にコマンドを入れて解析スタイルを切り替えることができます。次のコマンド・プロシージャは，現在の解析スタイルを保存し，解析スタイルを TRADITIONAL に設定し，コマンド (未指定) を実行して，保存された解析スタイルを復元します。

```
$ original_style= f$getjpi("","parse_style_perm")
$ SET PROCESS/PARSE_STYLE=TRADITIONAL
   .
   .
   .
$ SET PROCESS/PARSE_STYLE='original_style'
```

最初のコマンドにより， 'original_style' が現在の解析スタイルになります。 2 番目のコマンドにより，解析スタイルが TRADITIONAL に設定されます。最後のコマンドにより，解析スタイルが元のスタイルに再設定されます。

3.4.2 DCL コマンド・パラメータでの拡張ファイル名の使用

ファイル名をパラメータとして使用するコマンド・プロシージャは， ODS-5 環境では異なった結果を生成する場合があります。

解析スタイルは TRADITIONAL から EXTENDED に変更することができます。この項では， EXTENDED に変更した場合に影響を受ける可能性のある，次の分野について説明します。

- コマンド・プロシージャのファイル指定
- 大文字と小文字の区別および$FILE
- アンパサンドと一重引用符の置換

解析スタイルの切り替えの詳細については，第 3.4.1 項を参照してください。

3.4.3 コマンド・プロシージャのファイル指定

インダイレクト・コマンド・プロシージャが使用されている場合には，一部のプロシージャの引数を引用符で囲む必要がある場合があります。

次の例は，同じコマンド・ファイル SS.COM を使用した場合の， TRADITIONAL 解析スタイルと EXTENDED 解析スタイルでの出力の違いを示しています。

```
$ create ss.com
$ if p1 .nes. "" then write sys$output "p1 = ",p1
$ if p2 .nes. "" then write sys$output "p2 = ",p2
$ if p3 .nes. "" then write sys$output "p3 = ",p3
```

- 解析スタイルを TRADITIONAL に設定し，コマンド・ファイル SS.COM を実行すると，出力は次のようになる。

```
$ set process/parse_style=traditional
$ @ss ^ parg2 parg3
p1 = ^
p2 = PARG2
p3 = PARG3
```

サーカンフレックス (^) は，最初の引数であり (エスケープ文字ではない)，プロシージャ引数 p2 および p3 の大文字と小文字の区別が保存されていないことに注意してください。

- 解析スタイルを EXTENDED に設定し，同じコマンド・プロシージャを実行すると，出力は次のようになる。

```
$ set process/parse_style=extended
$ @ss ^ parg2 parg3
p1 = ^ PARG2
p2 = PARG3
```

コマンド・プロシージャでサーカンフレックス (^) が，スペースを引数の区切り文字としてではなくリテラル文字として識別するエスケープ文字として認識されていることと， "^ PARG2"が最初の引数であることに注意してください。大文字と小文字の区別は保存されません。

- サーカンフレックス (^) に引用符を追加すると，結果は次のようになる。

```
$ @ss "^" parg2 parg3
p1 = ^
p2 = PARG2
p3 = PARG3
```

  サーカンフレックス (^) は引用文字列の中にあるため，エスケープ文字として処理されません。

- 引数 p3 に引用符を追加すると，結果は次のようになる。

```
$ @ss "^" parg2 "parg3"
p1 = ^
p2 = PARG2
p3 = parg3
```

  プロシージャ引数 p3 の大文字と小文字の区別が保存されていることに注意してください。

- 解析スタイルを TRADITIONAL に設定すると，次のコマンドではサーカンフレックス (^) と， parg2 および parg3 の文字列とが，プロシージャ引数として処理され，コマンド・プロシージャを実行すると，結果は次のようになる。

```
$ set process/parse_style=traditional
$ @ss^ parg2 parg3
p1 = ^
p2 = PARG2
p3 = PARG3
```

- 解析スタイルを EXTENDED に設定すると，サーカンフレックス (^) は，スペースをリテラル文字として識別するエスケープ文字として処理される。 DCL は，ファイル "SS^_PARG2.COM"を探し，次の例で示されているエラーが生成される。

```
$ set process/parse_style=extended
$ @ss^ parg2 parg3
%DCL-E-OPENIN, error opening USER$DISK:[TEST]SS^_PARG2.COM; as input
-RMS-E-ACC, ACP file access failed
-SYSTEM-W-BADFILENAME, bad file name syntax
```

3.4.4 大文字と小文字の区別の保存と$FILE

DCL は，ファイル指定の大文字と小文字の区別を保存しようとします。実際にファイル指定の大文字と小文字の区別が保存されるのは，Command Definition Utility (CDU) を使用して定義されたコマンドに限られます。 DCL は，$FILE 解析タイプを持つコマンド定義ファイル (.CLD) の中で定義されている項目の大文字と小文字を保存します。

詳細については，『Command Definition Utility』を参照してください。

3.4.5 アンパサンドと一重引用符の置換

一重引用符に対してアンパサンド (&) による置換を使用して，従来型の解析の際に大文字と小文字の区別を保存することができます。

次の従来型の解析の例は，文字列の大文字と小文字の区別を変更する一連のコマンドを示しています。

```
$ set process/parse_style=traditional
$ x = "string"
$ define y 'x'
$ sho log y
   "Y" = "STRING" (LNM$PROCESS_TABLE)
$ define y &x
%DCL-I-SUPERSEDE, previous value of Y has been superseded
$ sho log y
   "Y" = "string" (LNM$PROCESS_TABLE)
```

アンパサンド (&) を使用することにより，変数 x に割り当てられた文字列の大文字と小文字の区別が保存されていることに注意してください。

一重引用符による置換は，コマンド行が大文字に設定される前に実行され，アンパサンドによる置換は，コマンド行が大文字に設定された後で実行されます。

次の拡張解析機能の例は，同じ一連のコマンドを示しています。

```
$ set process/parse_style=extended
$ define y 'x'
%DCL-I-SUPERSEDE, previous value of Y has been superseded
$ sho log y
   "Y" = "string" (LNM$PROCESS_TABLE)
$ define y &x
%DCL-I-SUPERSEDE, previous value of Y has been superseded
$ sho log y
   "Y" = "string" (LNM$PROCESS_TABLE)
```

変数 y に割り当てられた文字列はどちらも小文字で返されていることに注意してください。これは， DEFINE コマンドが，大文字と小文字の区別を保存する$FILE を使用するために発生します。

このような特徴を持つことから，アンパサンドによる置換は，解析スタイルが TRADITIONAL に設定されている場合でも EXTENDED ファイル名を指定するために使用することができます。次に例を示します。

```
$ set process/parse=extended
$ cre file^ name.doc
Contents of an ODS5 file
 Exit
```

```
$ set process/parse=traditional
$ a = "file^ name.doc"
$ type file^ name.doc
%DCL-W-PARMDEL, invalid parameter delimiter - check use of special characters
 \^NAME\
$ type 'a'
%DCL-W-PARMDEL, invalid parameter delimiter - check use of special characters
 \^NAME\
$ type &a
Contents of an ODS5 file
```

注意

アンパサンドの置換は，フォーリン・コマンドには使用できません。

3.5 DCL コマンドおよびユーティリティ

一部の DCL コマンドおよび OpenVMS ユーティリティは，拡張ファイル名のすべての機能を活用できるように変更されています。これらのユーティリティおよびコマンドは，エラーを発生させたり，目的の大文字と小文字の区別を修正することなく，拡張ファイル指定を受け付けることができるようになっています。

一方，拡張ファイル名を活用するための変更がほとんど加えられていない DCL コマンドおよび OpenVMS ユーティリティもあります。このようなユーティリティやコマンドは，拡張ファイル指定のほとんどの属性 (新しい文字や深いディレクトリ構造など) を正しく処理できると考えられています。

Extended File Specifications をサポートするための， DCL の新しい機能については，表 3–3 を参照してください。

第 2.1 節では， OpenVMS バージョン 7.2 で DCL コマンドおよび Open VMS ユーティリティによって提供されている，拡張ファイル名に関するさまざまなレベルのサポートについて，詳しく説明しています。

次の DCL コマンドおよび OpenVMS ユーティリティは，拡張ファイル名を完全にサポートしています。

ANALYZE /AUDIT
ANALYZE /DISK
ANALYZE /RMS
BACKUP
CONVERT
CONVERT /RECLAIM
COPY
CREATE /DIRECTORY
DELETE

DIRECTORY
DUMP
EDIT /ACL
EXCHANGE /NETWORK
FDL
PURGE
RECOVER/RMS
RENAME
SEARCH
SET SECURITY
SYSMAN
TYPE

表 3–3 は， Extended File Specifications をサポートするための， DCL の新しい機能を示しています。

表 3–3 DCL の新機能

DCL コマンド	新しい機能
COPY	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
DELETE	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
DIRECTORY	次の項目を追加。 • 新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子 /STYLE • クライアント属性を表示するための/FULL への表示項目
DUMP	次の項目を追加。 • 名前のタイプ属性を表示するための /DIRECTORY への表示項目 • 新しい属性を表示するための/HEADER への表示項目 • 新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子 /STYLE

(次ページに続く)

表 3–3 (続き) DCL の新機能

DCL コマンド	新しい機能
EXCHANGE NETWORK	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
F$FILE_ATTRIBUTES レキシカル関数	新しい項目コード FILE_LENGTH_HINT, VERLIMIT, DIRECTORY を追加。
F$GETDVI レキシカル関数	ACPTYPE 項目コードに新しいタイプを追加。
F$GETJPI レキシカル関数	新しい項目コード PARSE_STYLE_PERM および PARSE_STYLE_IMAGE を追加。
INITIALIZE	新しい修飾子/STRUCTURE=5 device-name[:] volume-label を追加。
PRINT	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
PURGE	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
RENAME	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SEARCH	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SET ACL	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SET DEFAULT	ODS-5 準拠のファイル指定を受け付けることができるように，ディレクトリ指定パラメータを変更。
SET DIRECTORY	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SET FILE	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SET PROCESS	キーワード EXPANDED および CONDENSED を持つ，新しい修飾子/PARSE_STYLE=(キーワード) を追加。
SET SECURITY	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。
SET VOLUME	新しい修飾子/STRUCTURE_LEVEL=5 を追加。
SHOW DEVICE/FULL	ディスク構造レベルを表示できるように，表示情報を更新。
SUBMIT	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。

(次ページに続く)

表 3–3 (続き) DCL の新機能

DCL コマンド	新しい機能
TYPE	新しいキーワード EXPANDED および CONDENSED を持つ，新しい修飾子/STYLE を追加。

Extended File Specifications をサポートするために OpenVMS オペレーティング・システムおよびそのユーティリティに追加された拡張機能の詳細については，『OpenVMS DCL ディクショナリ: A–M』，『OpenVMS DCL ディクショナリ: N–Z』，および『OpenVMS Utility Routines Manual』を参照してください。

3.6 拡張ファイル名の表示

表 3–3 に示されているように，一部の DCL コマンドでは，次のように新しい修飾子を使用して拡張ファイル名の表示を制御することができます。

```
/STYLE= [CONDENSED | EXPANDED]
```

この修飾子を使用すると，更新された DCL コマンドが拡張ファイル名およびそれらに関連するプロンプトを表示する方法を制御することができます。

キーワード CONDENSED を使用すると，多くのユーティリティで 255 バイトと定められている文字列の上限以内に納まるように生成されたファイル指定が表示されます。必要に応じて，このファイル指定には， DID 短縮形または FID 短縮形が含まれている場合があります。キーワード EXPANDED を使用すると，ディスク上に格納されているファイル指定が完全な形で表示され， DID 短縮形や FID は含まれません。

この後の項では， DIRECTORY コマンド， TYPE コマンド， PURGE コマンド，および DELETE コマンドに/STYLE 修飾子を使用した例が示されています。

3.6.1 DIRECTORY コマンド

DIRECTORY コマンドを使用すると，ディレクトリの内容を表示するときに，ファイル名を表示する形式を選択することができます。

```
DIRECTORY/STYLE=(keyword[,keyword])
```

DIRECTORY コマンドを使用すると，省略時の設定では，次の例のように，必要に応じて DID を使用し， DID が不要な場合には完全なディレクトリ指定に切り替わって，ファイル名が表示されます。

```
$ DIRECTORY

Directory TEST$ODS5:[23,1,0]

abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrs
tuvwxyABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxyABCDEFGHIJKLM
NOPQRSTUVWXY.abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYabcdef
ghijklmnopqrst;2

Total of 1 file.

Directory TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]

AddressFiles.DIR;1  LOGIN.COM;3         test.1;1       test^.1.clue;1
Travel.LIS;1        whee.;5       work.dat;8

Total of 8 files.

Grand total of 2 directories, 9 files.
```

DIRECTORY コマンドで/STYLE 修飾子を使用し，両方のキーワードを指定すると，2 列から成るディレクトリ・リストが表示されます。それぞれの列には，すべてのファイル名が含まれています。 CONDENSED の列には，必要に応じて DID および FID が含まれ， EXPANDED の列には，完全なディレクトリ名および完全なファイル名が含まれています。ファイル・エラーがあると， CONDENSED 列に表示されます。次の例は， DIRECTORY コマンドで/STYLE 修飾子を使用し，両方のキーワードを指定した結果を示しています。

```
$ DIRECTORY/STYLE=(CONDENSED,EXPANDED)

Directory TEST$ODS5:[23,1,0]        TEST$ODS5:[TEST.RANDOMTESTING.RANDO
                                    M]

abcdefghijklmnopqrstuvwxyABCDEFGHIJ abcdefghijklmnopqrstuvwxyABCDEFGHIJ
KLMNOPQRSTUVWXYabcdefghijklmnopqrst KLMNOPQRSTUVWXYabcdefghijklmnopqrst
uvwxyABCDEFGHIJKLMNOPQRSTUVWXYabcde uvwxyABCDEFGHIJKLMNOPQRSTUVWXYabcde
fghijklmnopqrstuvwxyABCDEFGHIJKLMNO fghijklmnopqrstuvwxyABCDEFGHIJKLMNO
PQRSTUVWXY.abcdefghijklmnopqrstuvwx PQRSTUVWXY.abcdefghijklmnopqrstuvwx
yABCDEFGHIJKLMNOPQRSTUVWXYabcdefghi yABCDEFGHIJKLMNOPQRSTUVWXYabcdefghi
jklmnopqrst;2                       jklmnopqrst;2
AddressFiles.DIR;1                  AddressFiles.DIR;1
LOGIN.COM;3                         LOGIN.COM;3
test.1;1                            test.1;1
test^.1.clue;1                      test^.1.clue;1
Travel.LIS;1                        Travel.LIS;1
whee.;5                             whee.;5
work.dat;8                          work.dat;8

Total of 8 files.
```

DIRECTORY コマンドでは，/STYLE 修飾子に一方または両方のキーワードを使用することができます。

3.6.2 TYPE コマンド

TYPE コマンドは，/STYLE 修飾子を受け付けます。この修飾子を使用すると，ファイルやプロンプトを入力するときにシステム・メッセージに表示されたファイル名形式を選択することができます。

```
$ TYPE/STYLE=(keyword)
```

次の例では， TYPE コマンドに修飾子 TYPE=EXPANDED および CONFIRM を使用しています。

```
$ TYPE/CONFIRM/STYLE=EXPANDED abc*.*rst;2
TYPE TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]abcdefghijklmnopqrstuvwxyzABCDEF
GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYabc
defghijklmnopqrstuvwxyzGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrst;2 ? [N]: Y
[System outputs contents of file]
```

3.6.3 DELETE コマンド

DELETE コマンドは/STYLE 修飾子を受け付けます。この修飾子を使用すると，コマンドを実行するときに表示するファイル名形式を選択することができます。

```
$DELETE/STYLE=(keyword)
```

次の例の反復記号 (...) は，ファイル名の中に多くの文字が含まれていることを示します。これらの例では， CONFIRM 修飾子を使用して，システム・メッセージを生成しています。

省略時の値 (CONDENSED) を使用した DELETE:

```
$ DELETE/CONFIRM abc*.*.*
DELETE TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]abcAlphabet.stuff;1 ? [N]: Y
DELETE TEST$ODS5:[23,1,0] abcdefg. . .QRSTUVWXY.abcdefg. . .tuvw
xy;1 ? [N]: Y
```

完全なファイル指定が必要な場合には， DELETE コマンドで/STYLE 修飾子と EXPANDED キーワードを使用します。

```
$ DELETE/CONFIRM/STYLE=EXPANDED abc*.*.*
DELETE TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]abcAlphabet.stuff;1 ? [N]: Y
DELETE TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]abcdefg. . .QRSTUVWX
Y.abcdefg. . .tuvwxy;1 ? [N]: Y
```

3.6.4 PURGE コマンド

PURGE コマンドは/STYLE 修飾子を受け付けます。この修飾子を使用すると，コマンドを実行するときに表示するファイル名形式を選択することができます。

```
$ PURGE/STYLE=(keyword)
```

次の例の反復記号 (...) は，ファイル名の中に多くの文字が含まれていることを示します。これらの例では， CONFIRM 修飾子を使用して，システム・メッセージを生成しています。

省略時の値 (CONDENSED) を使用した PURGE:

```
$ PURGE/CONFIRM
DELETE TEST$ODS5:[23,1,0]abcdefg. . .QRSTUVWXY.abcdefg. . .tuvwxy;1
? [N]: Y
```

完全なファイル指定が必要な場合には， PURGE コマンドで/STYLE 修飾子と EXPANDED キーワードを使用します。

```
$ PURGE/CONFIRM/STYLE=EXPANDED
DELETE TEST$ODS5:[TEST.RANDOMTESTING.RANDOM]abcdefg. . .QRSTUVWXY.ab
cdefg. . .tuvwxy;1 ? [N]: Y
```

3.7 拡張ファイル名の端末表示

端末に拡張ファイル名を表示するには，端末で表示する文字セットとして ISO Latin-1 を指定しなければなりません。このように設定しないと，画面に表示される文字は， PC で表示される文字と一致しなくなります。図 C–1 には， DEC MCS と ISO Latin-1 文字セットの間で異なる文字のリストが示されています。

ISO Latin-1 文字セットを DECterm 上で正しく表示するには， Options メニューの General サブメニューからUPSS ISO Latin 1を選択します。

DEC で定義している文字セットを DECterm 上で正しく表示するには， Options メニューの General サブメニューからUPSS DEC Supplementalを選択します。

ISO Latin-1 文字セットを VT320 または VT420 上で正しく表示するには， Setup メニューの General サブメニューからUPSS ISO Latin 1を選択します。

DEC で定義している文字セットを VT320 または VT420 上で正しく表示するには，Setup メニューの General サブメニューからUPSS DEC Supplementalを選択します。

4

OpenVMS アプリケーション開発での拡張ファイル名に関する注意点

この章では， Extended File Specifications に関するアプリケーションのサポートを評価する方法について説明します。

4.1 現在のサポート状態の評価

OpenVMS Alpha バージョン 7.2 のテストの一環として， OpenVMS アプリケーションの開発者は，すべての既存のアプリケーションを評価し，テストして， Extended File Specifications の現在のサポート・レベルと，そのレベルが適切であるかどうかを確認する必要があります。サポートのレベルについては，第 2.1 節を参照してください。

文書化されていないインタフェースを使用してコーディングされているアプリケーションでは，深いディレクトリまたは拡張ファイル名のサポートが提供されないことがあります。第 4.1.2 項では，アプリケーションが拡張ファイル名をサポートできなくなるアプリケーション属性が示されています。第 4.1.3 項では，アプリケーションが ODS-5 ボリュームをサポートできなくなるアプリケーション属性が示されています。

これらのアプリケーションが Extended File Specifications をサポートするように変更するか， Extended File Specifications ではこれらを使用しないようにするか，どちらかにします。 Extended File Specifications の省略時のサポートを提供するようにアプリケーションを変更する方法については，第 4.2.1 項を参照してください。完全サポートを行うようにアプリケーションをアップグレードする方法については，第 4.2.2 項を参照してください。

4.1.1 省略時のサポート

変更されていない OpenVMS アプリケーションのほとんどは，省略時のサポートのカテゴリに分類されます。特に，このようなアプリケーションは， RMS 呼び出しを実行する際に，新しい API ではなく従来の API を使用します (新しい RMS API の詳細については，第 B.2 節を参照してください)。高水準言語呼び出しを使用してファイル操作を実行するアプリケーションも，言語の実行時ライブラリが完全サポートに変更されていない限り，このカテゴリに分類されます。[1]ほとんどの場合は，これらのアプリケーションを変更しなくても， Extended File Specifications の環境で正常に動作します。

[1] OpenVMS バージョン 7.2 の時点では，完全サポートにアップグレードされた言語の RTL はありません。

4.1.2 Extended File Names の非サポート

次のいずれかに該当するアプリケーションでは，拡張ファイル名がサポートされていないことがあります。

1. QIO インタフェースを使用してファイル名を指定している。開発者は，すべてのレイヤード・プロダクトおよびアプリケーションを調べ， RMS インタフェースと XQP インタフェースとの間での相互作用を評価する必要がある。拡張ファイル名の形式は，それぞれのインタフェースによって異なるため，アプリケーションが RMS と XQP の両方で同じファイル名を使用できるとは限らない。さらに， XQP では，更新されていないアプリケーションが拡張ファイル名を使用することができない。拡張ファイル名をサポートするために XQP に追加された変更の詳細については，第 B.3 節を参照。有効なファイル名は，インタフェースによって異なることがある。
2. 区切り記号や有効な文字の位置の位置など，ファイル指定の構文に関する仮定がある。
3. ファイル指定の大文字と小文字の区別に仮定がある。 RMS は，大文字と小文字が混在しているファイル指定や，小文字だけのファイル指定をすべて大文字に変換する処理を実行しないため，文字列のマッチング処理に影響を与える可能性がある。
4. 従来のディレクトリの深さに依存している (8 レベル未満)。

4.1.3 ODS-5 ボリュームの非サポート

ディレクトリの内容やファイル・ヘッダのデータのディスクでの構造など，ファイル・システムの内部知識を使用するアプリケーションは， ODS-5 ボリュームでは正常に動作しません。

4.2 Extended File Specifications サポートのためのアプリケーションのアップグレード

この後の項では Extended File Specifications のサポート・レベルをアップグレードするために必要な変更について説明します。アプリケーションを完全サポート・レベルにアップグレードするには，最初にそのアプリケーションが省略時のサポート・レベルを満たしていなければならないことに注意してください。

注意

RMS インタフェースや QIO インタフェースを使用せずにディスク入出力を実行している場合，現在のアプリケーションの Extended File Specifications のサポート・レベルは，現在使用しているインタフェース (言語の実行時ライブラリなど) が完全サポートを提供しているかどうかによって決まります。

4.2.1 省略時サポートへのアップグレード

Extended File Specifications の省略時サポートを提供するようにアプリケーションをアップグレードするには，4.2.1.1 および 4.2.1.2 でそれぞれ推奨されているように，少なくともそのアプリケーションが ODS-5 ボリューム構造と拡張ファイル命名機能の両方をサポートしている必要があります。省略時サポートについては，第 2.1.2 項で説明しています。

4.2.1.1 ODS-5 サポートの提供

新しい ODS-5 ボリューム構造をサポートしていないアプリケーションは，従来型のファイル指定だけを処理した場合でも，これらのボリューム上では正常に動作しません。

ODS-5 ボリューム上でアプリケーションが正常に動作しない場合には，そのアプリケーションについて次の点を確認してください。

- **ODS-5 ボリュームにアクセスする際，このアプリケーションは物理入出力または論理入出力のどちらを使用してファイル・システムをバイパスしているのか? あるいは，BITMAP.SYS のようなメタデータ・ファイルに直接アクセスしているのか?**

  このようなアプリケーションは，通常，ディスク・デフラグメンタのようなシステム・プログラムか，またはディスクに直接アクセスすることによってオーバヘッドを避けようとしているプログラムです。このようなアプリケーションは，ディスク上のファイルまたはディレクトリに関する特定の知識に依存していますが，この知識は， ODS-5 構造を導入したことによって，すでに変更されています。

  推奨事項:アプリケーションには，できる限り文書化されたインタフェースや構造を使用します。

- **このアプリケーションは，ディレクトリ・ファイルに直接アクセスし，内容を解釈しているか?**

  そのような場合，拡張ファイル名が含まれているディレクトリをこのアプリケーションが処理するときにエラーが発生する可能性があります。

  推奨事項: RMS[2]インタフェースまたはQIO インタフェース，あるいは LIB$FIND_FILE のような LIBRTL ルーチンで提供されている検索関数を使用できるようにアプリケーションを変更します。

[2] OpenVMS Alpha バージョン 7.2 では， RMS ディレクトリのキャッシング・サイズが大幅に増加したため，大規模なディレクトリでの$SEARCH システム・サービスの性能が飛躍的に向上しました。

4.2.1.2 拡張ファイル命名機能サポートの提供

アプリケーションが拡張名を正しく処理できない場合には，そのアプリケーションについて次の点を確認してください。

- **このアプリケーションは，ファイル指定の構文を解析したり，その構文に関する知識を仮定して動作しようとしているか**?

  たとえば，アプリケーションがディレクトリ指定の先頭を探すために大括弧 ([) を検索したり，ファイル指定の末尾を示すスペース文字を探している場合があります。

  推奨事項:アプリケーションは，ファイル指定が有効かどうかを判断するためには，実際の名前を使用して予備テストを実行するのではなく， RMS に依存するべきです。 NAM ブロックの Use the NAM$L_NODE フィールド， NAM$L_DEV フィールド， NAM$L_DIR フィールド， NAM$L_TYPE フィールド，および NAM$L_VER フィールドを使用するか，または SYS$FILESCAN を使用して，この情報を取り出します。

- **このアプリケーションは，文字列比較演算を実行することによって，2 つのファイルが同一であるかどうかを判断しようとしているか**?

  ファイル名の大文字と小文字は区別されず，同じ文字を表す方法が複数あるため， 2 つの文字列が同一のファイルを表している場合でも，文字列比較演算を実行するとエラーが発生する可能性があります。

  推奨事項:新しいシステム・サービス$CVT_FILENAMES を使用してファイル名を比較する例については，サンプル・プログラム[SYSHLP.EXAMPLES]FILENAME_COMPARE.Cを参照してください。

- **このアプリケーションは，** NAM$L_FNB **フィールドにある** NAM$V_DIR_LVLS **ビットに依存して，現在のファイル指定にあるディレクトリ・レベルの数を判断しているか**?

  このフィールドには 3 ビットしかないため，指定できるのは最大で 8 レベルまでです。アプリケーションがこれらのビットを使用することはほとんどなく，通常，これらのビットは NAM が相対ファイル指定として指定されるときに， RMS によって使用されます。

  推奨事項: OpenVMS バージョン 7.2 以降， NAM ブロックと NAML ブロックの両方で，新しくより大きいフィールド NAM$W_LONG_DIR_LEVELS を使用することができます。ディレクトリ・レベルの正しい数を知るには，このフィールドを使用します。

- **このアプリケーションは，** NAM$V_WILD_UFD **および** SFD1 - SFD7 **ビットに依存して，ワイルドカード・ディレクトリがあるかどうかを判断しているか**?

  このようなビットは 8 つしかないため，報告できるのは最初の 8 ディレクトリ・レベルのワイルドカードに限られます。アプリケーションがこれらのビットを使用することはほとんどなく，通常，これらのビットは NAM が相対ファイル指定として指定されるときに， RMS によって使用されます。

推奨事項: OpenVMS バージョン 7.2 以降, NAM ブロックと NAML ブロックの両方で，新しくいフィールド NAML$W_FIRST_WILD_DIR を使用することができます。ワイルドカードが含まれている最も上のディレクトリ・レベルを知るには，このフィールドを使用します。

- **このアプリケーションは，ファイル・システムに対し QIO インタフェースを使用して，QIO のファイル名を直接指定したり要求しているか**?

  QIO インタフェースでは，拡張ファイル名を受け付けたり返すためには，アプリケーションが拡張ファイル名を認識できることを明示的に指定する必要があります。さらに，拡張ファイル名のファイル名形式は, RMS インタフェースと QIO インタフェースとでは異なっています。しかも，一部のファイル名は, 2 バイトの Unicode (UCS-2) 文字で指定されていることがあります。アプリケーションは, 2 バイトを占有する 1 文字を処理できるようになっていなければなりません。

  推奨事項: QIO インタフェースを使用しているほとんどのアプリケーションは，RMS を使用してファイル指定を解析したり，ファイルのファイル ID やディレクトリ ID を取り出している。そのようなアプリケーションは，これらの ID の値を使用して, QIO インタフェースからファイルにアクセスします。このアクセス方式は，拡張名にも使用できます。この方式に変更することによって，問題を解決できます。

  NAML ブロックの NAML$L_FILESYS_NAME フィールドから QIO システムが使用する名前を取得したり，新しいシステム・サービス (SYS$CVT_FILENAME) を使用して, RMS ファイル名と QIO ファイル名との間で相互に変換することもできます。この場合には, QIO サービスで拡張 FIB ブロックを使用して，アプリケーションが拡張名を認識していることを指定し，バッファを最大サイズまで拡大し, 2 バイトの Unicode 文字を処理できるように準備する必要があります。

4.2.2 完全サポートへのアップグレード

システム管理ユーティリティやディスク管理ユーティリティなど一部の OpenVMS アプリケーションでは, Extended File Specifications の完全サポートが必要です。通常，このようなユーティリティは, DID や FID の短縮形を持たないすべてのファイル指定を表示し，操作しなければなりません。 Extended File Specifications のすべての機能を完全にサポートするようにアプリケーションをアップグレードするには，次の操作を行います。

1. RMS NAM ブロックの使用部分をすべて新しい NAML ブロックに変換します。
2. RMS で使用している入力ファイル名バッファと出力ファイル名バッファを拡張します。このためには，短いバッファ・ポインタ (NAML$L_ESA と NAML$L_RSA) ではなく, NAML の拡張バッファ・ポインタと結果バッファ・ポインタ (NAML$L_LONG_EXPAND と NAML$L_LONG_RESULT) を使用し，バッファ・サイズを NAM$C_MAXRSS から NAML$C_MAXRSS に増やします。

3. FAB ファイル名のバッファ・フィールド (FAB$L_FNA) に長い (255 バイトを超える) ファイル名が指定されている場合には， NAML のファイル名バッファ・フィールド (NAML$L_LONG_FILENAME) を代わりに使用します。 FAB の省略時の名前バッファ・フィールドに (FAB$L_DNA) 長いファイル名が指定されている場合には， NAML の省略時の名前バッファ・フィールド (NAML$L_LONG_DEFNAME) を代わりに使用します。
4. LIB$FIND_FILE ルーチン， LIB$RENAME ルーチン，または LIB$DELETE ルーチンを使用する場合には，flags引数に LIB$M_FIL_LONG_NAMES を設定します (flagsは， LIB$DELETE ルーチンの新しい引数です)。 NAML ブロックを NAM ブロックの代わりに使用すると，他に変更を加えずに情報を LIB$FILE_SCAN に渡すことができることに注意してください。
5. LIB$FID_TO_NAME ルーチンを使用する場合には，返されるファイル指定の記述子を変更して， 4095 (NAML$C_MAXRSS) バイトに増加された最大バイト数に対応するよう変更しなければならない可能性があります。
6. FDL$CREATE ルーチン， FDL$GENERATE ルーチン， FDL$PARSE ルーチンまたは FDL$RELEASE ルーチンを使用する場合には，flags引数に FDL$M_LONG_NAMES を設定しなければなりません。
7. ソース・コードを調べて，ファイル指定が 8 ビットで 255 文字を超えないという仮定が内部的に設定されていないことを確認します。

A

ユーザを対象とした Extended File Specifications の注意点

Extended File Specifications により，ユーザは Windows スタイルのファイル指定を OpenVMS 環境で使用できるようになります。ユーザが Extended File Specifications の環境になじめるようにする方法の 1 つに， ODS-2 と ODS-5 で見られるファイル名のいくつかの相違点について説明することがあります。相違点としては， ODS-2 から ODS-5 のスタイルに切り替えた場合に最も目立つものを取り上げます。

この後の節では，システム管理者がユーザに知らせておくと効果的であると思われる使用上の注意を示します。これらの注意点は，次のように分類されます。

- Extended File Specifications の新しい特性
- ODS-2 と ODS-5 の同時使用
- アーキテクチャに関する注意点

A.1 Extended File Specifications の新しい特性

この節では，ユーザが初めて使用する Extended File Specifications の新しい特性に関する注意事項について説明しています。

◆ボリューム構造を意識すること

ODS-5 ファイルを ODS-5 ボリュームに格納できるように，ディスクが ODS-2 ボリュームであるか，または ODS-5 ボリュームであるかを把握しておく必要があります。

ボリューム・タイプは，次のようなコマンドを実行することによって表示できます。

```
$ SHOW DEVICE DKA500:/FULL

  Disk AABOUT$DKA500:, device type RZ25, is online, allocated, deallocate
  on dismount, mounted, file-oriented device, shareable.

    Error count                    0    Operations completed 155
    .
    .
    .
  Volume Status:  ODS-5, subject to mount verification, file high-water
  marking, write-back caching enabled.

$ SHOW DEVICE DKA200:/FULL

  Disk AABOUT$DKA200:, device type RZ25, is online, allocated, deallocate
  on dismount, mounted, file-oriented device, shareable.
```

```
   Error count                    0    Operations completed 232
   .
   .
   .
  Volume Status:  ODS-2, subject to mount verification, file high-water
  marking, write-back caching enabled.
```

それぞれのコマンドの後に表示される“Volume Status:”は，ボリュームが ODS-5 であるか，または ODS-2 であるかを示しています。

◆ ODS-2 ボリューム上では拡張ファイル名を使用しない

ODS-2 ボリューム上では，拡張ファイル名を持つファイルを作成することができません。

次の例では， DKA200 が ODS-2 ボリューム，解析スタイルが EXTENDED であるため， RMS は，エラー・メッセージを返します。

```
$ SET DEFAULT DKA200:[TEST]
$ CREATE x.x.x.x
%CREATE-E-OPENOUT, error opening DAK200:[TEST]X^.X^.X.X; as output
-RMS-E-CRE, ACP file create failed
-SYSTEM-W-BADFILEVER, bad file version number
```

◆大文字と小文字の区別は拡張ファイル名を最初に作成したときに決まる

ODS-5 ボリューム上では， 1 つのファイルのすべてのバージョンで，大文字と小文字の区別が同じです。つまり，大文字と小文字の区別は，そのファイル名が最初に作成されたときのまま保存されます。

次の例では， DKA500 は ODS-5 ディスクです。

```
$ SET DEFAULT DKA500:[TEST]
$ SET PROCESS /PARSE_STYLE=EXTENDED
$ CREATE myfile.txt
Ctrl/Z
$ CREATE MYFILE.TXT
Ctrl/Z
$ DIRECTORY

Directory DKA500:[TEST]

myfile.txt;2        myfile.txt;1
```

◆ Extended File Specifications の大文字と小文字の区別の保存とその無視に注意する

ODS-5 ボリュームでは，ファイルが最初に入力されたときの大文字と小文字の区別が保存されますが，ファイルの検索は大文字と小文字を区別せずに実行されます。同様に，ユーザがコマンド・プロシージャの中で.EQS. や F$LOCATE などの DCL 文字関数を使用するときなどに比較を実行する場合にも，注意しなければなりません。

次の例は， DCL の中で大文字と小文字を区別しないファイル名のマッチングの重要さを示しています。このプログラムでは，大文字と小文字を区別するマッチングを実行する場合には引数を指定せず，大文字と小文字を区別しないマッチングを実行する場合には引数を指定することに注意してください。

このプログラムでは， F$SEARCH を使用して，".TXT"というファイル・タイプのすべてのファイルを見つけます。 RMS (したがって F$SEARCH) は大文字と小文字を区別しないマッチングを実行するため，".txt"というファイル・タイプのファイルも見つけます。次に， F$SEARCH は F$LOCATE を使用して， "TEST"という名前を持つファイル名を検索します。 F$LOCATE は大文字と小文字を区別するマッチングを実行するため，あらかじめ文字列を大文字に変換しておかないと，マッチングは失敗します。

```
$ case_blind = 0
$ if p1 .nes. "" then case_blind = 1 1
$loop:
$  file = f$search("*.TXT;") 2
$  if file .eqs. "" then goto not_found
$  write sys$output "Search returns " + file
$  if case_blind .eq. 1 then file = f$edit(file,"UPCASE") 3
$  if (f$locate("TEST",file) .ne. f$length(file)) then goto found 4
$  goto loop
$found:
$   write sys$output "Found a file matching TEST"
$   exit
$not_found:
$   write sys$output "Did not find file matching TEST"
$   exit
```

次に，この例の中で番号が付いている部分について説明します。

1 引数 (大文字と小文字を区別しない比較演算を実行するよう，プログラムに要求する) がある場合には，"case_blind"を 1 に設定する。

2 末尾が".TXT"または".txt"になっているファイルを取得する (F$SEARCH は大文字と小文字を区別しないため)。

3 手順 1 で大文字と小文字を区別しない比較演算が選択された場合には，ファイル名を大文字に変更して，大文字と小文字を区別しない比較演算を実行する。

4 F$LOCATE は，ファイルを見つけると，"found:"に進む。

次の例では，検索プログラムは大文字と小文字を区別する検索を実行し，マッチするものを見つけていません。

```
$ @test
Search returns DKA300:[FISHER]test.txt;1
Did not find file matching TEST
```

次の例では，検索プログラムは大文字と小文字を区別しない検索を実行し，マッチするものを見つけません。

```
$ @test case-blind
Search returns DKA300:[FISHER]test.txt;1

Found a file matching TEST
```

◆ CONDENSED スタイルのファイル名では短縮形と完全なディレクトリが別々に表示される

一部のシステムユーティリティおよび DIRECTORY のような DCL コマンドには，ファイル名を表示する方法を制御するスタイル・スイッチがあります。スタイルが CONDENSED の場合には，最長で 255 バイトのファイル名が表示される。ファイル指定が 255 バイトの上限に達すると，ディレクトリ名はディレクトリ ID (DID) に短縮される。

DIRECTORY コマンドは， DID の短縮形によるディレクトリ名を短縮されていないディレクトリ名と区別するため，短縮が行われると，それぞれのディレクトリ名に対して個別のヘッダを作成します。

```
$ DIR/STYLE=CONDENSED

Directory DKA300:[DEEPER.aaaa.bbbb.cccc.dddd.eeee.ffff.gggg.hhhh.iiii._ten.aaaa.
bbbb.cccc.dddd.eeee.ffff.gggg.hhhh.iiii._ten.aaaa.bbbb.cccc.dddd.eeee.ffff.gggg.
hhhh.iiii._ten.aaaa.bbbb.cccc.dddd.eeee.ffff.gggg.hhhh.iiii._ten]1

aaaa.txt;1

Total of 1 file.

Directory DKA300:[528,7036,0]2

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.txt;1

Total of 1 file.

Grand total of 2 directories, 2 files.3
```

1 CONDENSED スタイルでは，ディレクトリ名とファイル名の組み合わせが 255 バイトを超えない場合には，ディレクトリ名は DID に短縮されない。

2 CONDENSED スタイルでは，ディレクトリ名とファイル名の組み合わせが 255 バイトを超える場合には，ディレクトリ名は DID に短縮される。

3 同じ 1 つのディレクトリに対して完全なディレクトリ形式と短縮形によるディレクトリ形式の両方を表示する DIRECTORY コマンドを実行すると，DIRECTORY はこれらを 2 つの異なるディレクトリとしてカウントする。

DIRECTORY コマンドの詳細については，『OpenVMS DCL ディクショナリ』を参照してください。

◆等価名としての Extended File Specifications に注意する

Extended File Specifications のエスケープ文字，サーカンフレックス (^) は，論理名の等価名文字列の中では使用されません。エスケープ文字を必要とする拡張ファイル名に対して論理名を定義するときには， DEFINE コマンドの中の拡張ファイル名にエスケープ文字を使用しないようにします。次に例を示します。

```
$ define xxx a&b
$ dir xxx

Directory DKA500:[EXTENDED_FILES]

a^&b.txt;1
```

```
Total of 1 file.
```

A.2 ODS-2 と ODS-5 の同時使用

この節では，クラスタ内での ODS-2 と ODS-5 の同時使用に関する注意事項について説明します。

◆ボリュームが混在している環境では従来型のファイル名を使用する

ODS-2 ボリュームと ODS-5 ボリュームの両方を使用している場合には， ODS-2 のファイル名と ODS-5 のファイル名の互換性の問題が発生することを避け， OpenVMS の以前のバージョンとの下位互換性を保つために，従来型の ODS-2 のファイル名だけを使用します。

◆エラー・メッセージは解析スタイルによって異なることがある

ユーザに対して表示されるエラー・メッセージは，解析スタイルによって異なることがあります。解析スタイルが EXTENDED に設定されていると，以前は DCL レベルで検出されていた構文エラーが， RMS や XQP などのファイル・システムのレベルに渡されます。このため，ファイルの構文エラーに対してユーザが受け取るメッセージは，解析スタイルとボリューム構造によって多少異なることがあります。

次にさまざまなエラー・メッセージの例を示します。

- ODS-5 ボリューム上での TRADITIONAL スタイルと EXTENDED スタイルの例

```
$ SHOW DEVICE DKA500:/FULL

  Disk AABOUT$DKA500:, device type RZ25, is online, allocated, deallocate
  on dismount, mounted, file-oriented device, shareable.

    Error count                    0    Operations completed 155
    .
    .
    .
  Volume Status:  ODS-5, 1  subject to mount verification, file high-water
  marking, write-back caching enabled.

$ SET PROCESS /PARSE_STYLE=TRADITIONAL 2
$ OPEN /WRITE FILE z.z.z.z
%DCL-W-PARMDEL, invalid parameter delimiter - check use of special
characters \.Z\ 3
$ SET PROCESS /PARSE_STYLE=EXTENDED 4
$ OPEN /WRITE FILE z.z.z.z
$ 5
```

1 ボリュームは ODS-5。

2 解析スタイルを TRADITIONAL に設定する。

3 DCL は一部の ODS-5 ファイル名に対してこのようなエラーを返す。

4 解析スタイルを EXTENDED に設定する。

5 DCL はファイルを作成する。

- ODS-2 ボリューム上での TRADITIONAL スタイルと EXTENDED スタイルの例

```
  Disk AABOUT$DKA200:, device type RZ25, is online, allocated, deallocate
  on dismount, mounted, file-oriented device, shareable.

    Error count                    0    Operations completed 232
    .
    .
    .
  Volume Status:  ODS-2, 1  subject to mount verification, file high-water
  marking, write-back caching enabled.

$ SET PROCESS /PARSE_STYLE=TRADITIONAL 2
$ OPEN /WRITE FILE z.z.z.z
%DCL-W-PARMDEL, invalid parameter delimiter - check use of special
characters \.Z\ 3
$ SET PROCESS /PARSE_STYLE=EXTENDED 4
$ OPEN /WRITE FILE z.z.z.z
%DCL-E-OPENIN, error opening
-RMS-E-CRE, ACP file create failed 5
-SYSTEM-W-BADFILEVER, bad file version number
```

1 ボリュームは ODS-2。

2 解析スタイルを TRADITIONAL に設定する。

3 DCL はエラー・メッセージを返す。

4 解析スタイルを EXTENDED に設定する。

5 DCL はこのファイル名を受け付けるが, XQP はエラーを返す。

- 同じ構文エラーに対するエラー・メッセージが異なる例

```
$ SHOW DEVICE DKA500:/FULL

  Disk AABOUT$DKA500:, device type RZ25, is online, allocated, deallocate
  on dismount, mounted, file-oriented device, shareable.

    Error count                    0    Operations completed 155
    .
    .
    .
  Volume Status:  ODS-5, 1  subject to mount verification, file high-water
  marking, write-back caching enabled.

$ SET PROCESS /PARSE_STYLE=TRADITIONAL 2
$ CREATE a^<b.c

%DCL-W-PARMDEL, invalid parameter delimiter - check use of special
characters
 \^\ 3

$ SET PROCESS /PARSE_STYLE=EXTENDED 4
$ CREATE a^<b.c

%CREATE-E-OPENOUT, error opening a^<b.c as output
-RMS-F-SYN, file specification syntax error 5
```

1 ボリュームは ODS-5。

2. 解析スタイルを TRADITIONAL に設定する。
3. DCL は構文エラーに対するエラー・メッセージを返す。
4. 解析スタイルを EXTENDED に設定する。
5. RMS は，同じ構文エラーに対して異なるエラー・メッセージを返す ("<"は，拡張ファイル名では使用できない)。

◆暗黙のファイル名出力に注意する

処理されるファイルを基にユーティリティが出力ファイルを作成できるようにする場合には，省略時の値に注意が必要です。誤って ODS-2 ボリュームに拡張ファイル名を格納しようとすることがないように，ファイルが格納される場所を確認する必要があります。

次に，予想しなかった場所にファイルが格納される例をいくつか示します。

- アプリケーションまたはユーティリティが ODS-5 の拡張ファイル名を ODS-2 (DKA200:) ボリュームに書き込もうとすると，次の例のようにエラーが発生する。

```
$ SHOW DEFAULT
  DKA200:[DOREO]
$ DUMP /OUTPUT DKA500:[DOREO]This^_is^_a^_file.Dat
%DUMP-E-OPENOUT, error opening DKA200:[DOREO]THIS^_IS^_A^_FILE.DMP;as
output
-RMS-E-CRE, ACP file create failed
-SYSTEM-W-BADFILENAME, bad file name syntax
```

/OUTPUT 修飾子を使用して指定した出力ファイルは，省略時の設定では，入力ファイルと同じ名前でファイル・タイプが.DMP になり省略時のディレクトリ内に生成されます。入力ファイル指定は ODS-5 ボリューム上の拡張名ですが，.DMP ファイルは ODS-2 ボリュームに書き込まれるため，従来型の名前でなければなりません。このため，エラーが発生します。

- 次の条件のすべてが該当すると，バッチ・コマンド・ファイルの実行は失敗する。
  - ログ・ファイルが暗黙にまたは明示的に要求された。
  - 暗黙のまたは明示的なログ・ファイル指定に拡張ファイル名が含まれている (つまり，このログ・ファイルの名前は ODS-2 に準拠していない)。
  - ログ・ファイルが ODS-2 ボリューム上に作成された。

  このような場合，ログ・ファイルを作成できないため，バッチ・コマンド・ファイルの実行は失敗します。このような状況は，論理名 SYS$LOGIN が ODS-2 ボリュームを参照したときに最も多く発生します。これは，ログ・ファイルが SYS$LOGIN デバイス上で暗黙に作成されるためです。さらに，通知が無効になっていると，バッチ・ジョブが実行されなかったことが通知されません。

このようなエラーを防ぐには，拡張ファイル名を持つコマンド・ファイルを実行するときには，/LOG=修飾子および ODS-2 準拠のログ・ファイル指定を使用するようにします。

A.3 アーキテクチャに関する注意点

この節では，システム・アーキテクチャに関する Extended File Specifications の注意事項について説明します。

◆ VAX システムでは拡張ファイル名を表示できない

VAX システムで ODS-5 ボリュームをマウントすることはできますが， VAX システムにログインしても，拡張ファイル名を表示することができません。その代わりに，疑似名が表示されます。

- VAX 上では， 2 バイトの Unicode 文字列が含まれているファイル名を表示しようとすると，\ PUNICODE \ .??? という疑似名が表示されます。
- 有効な ODS-2 名でない他の名前はどれも，\ PISO_LATIN \ .??? と表示されます。

次の例は，同じディレクトリ名が Alpha システムと VAX システムとで表示される場合の違いを示しています。

- Alpha システム

```
$ DIRECTORY DPA100:[TEST]
Directory DPA100:[TEST]
Accounting^_data.lis;1                 atest.txt;1
```

- VAX システム

```
$ DIRECTORY DPA200:[TEST]
Directory DPA200:[TEST]
\PISO_LATIN\.???                       ATEST.TXT
```

さらに， VAX システムでのディレクトリの深さは， 8 (最上位論理名を使用した場合には 16) に制限されます。

A.4 制約事項

この節では，拡張ファイル名を使用する場合の制約事項について説明します。

◆チルダ (~) をファイル名の最初の文字として使用しない

DEC C Run Time Library (CRTL) を使用することにより，プログラマは， creat()やfopen()などのルーチンに対して， UNIX スタイルと VMS スタイルの両方のファイル指定を実行することができます。

UNIX のファイル指定では，最初の文字がチルダ (~) になっているパス名は，ユーザのホーム・ディレクトリを表します。一方， OpenVMS の拡張ファイル名では，チルダはファイル名またはディレクトリ名のどの位置にあっても有効です。

下位互換性を保つために， CRTL では，引き続きファイル名やディレクトリ名の先頭にあるチルダ (~) をユーザのホーム・ディレクトリを表します。先頭がチルダ (~) になっている OpenVMS ファイル名を， UNIX スタイルのファイル指定を受け付ける CRTL ルーチンに渡すには，^~のように，チルダの前にエスケープ文字としてサーカンフレックス (^) を付けます。

次の DEC CRTL 関数は OpenVMS の拡張ファイル名を受け付けますが，ファイル指定の中で先頭にチルダ (~) がある場合にはこの構文が必要になります。

.create
.fopen
.freopen
.open
.stat

B

技術情報

この付録では，本書の他の章で登場した技術情報を，まとめて示しています。

B.1 システム・サービスの変更点

この節では，以下のシステム・サービスについて説明します。

- 新しいサービス
  - $SET_PROCESS_PROPERTIESW
  - $CVT_FILENAME
- 変更されたサービス
  - $CREPRC
  - $GETJPI
  - $SETDDIR

B.1.1 $SET_PROCESS_PROPERTIESW システム・サービス (Alpha システムのみ)

$SET_PROCESS_PROPERTIESW システム・サービスは，プロセスに関連付けられる単純な値を設定します。

形式

$SET_PROCESS_PROPERTIESW mbz1 ,mbz2 ,mbz3 ,property ,value, prev_value

C プロトタイプ

```
int sys$set_process_properties(
        unsigned int mbz1,
        unsigned int mbz2,
        unsigned int mbz3,
        unsigned int property,
        unsigned __int64 value,
        unsigned __int64 *prev_value);
```

引数
mbz1,mbz2,mbz3

Compaq で将来使用するために予約されています。 0 が指定されていなければなりません。

property
OpenVMS usage: integer
type: longword (unsigned)
access: read only
mechanism: by value

設定するプロパティを選択する定数。

プロパティの有効な値は，$PPROPDEF マクロによって次のように定義されています。

プロパティ・コード	説明
PPROP$C_PARSE_STYLE_TEMP:	使用するコマンド解析のタイプ。この値は，イメージの存在期間に限って設定される。イメージが存在しなくなると，この値は恒久的なスタイルに戻る。有効な値は， PARSE_STYLE$C_TRADITIONAL および PARSE_STYLE$C_EXTENDED。
PPROP$C_PARSE_STYLE_PERM:	使用するコマンド解析のタイプ。この値は，スタイルが再び設定されない限り，プロセスの存在期間中は有効である。有効な値は， PARSE_STYLE$C_TRADITIONAL および PARSE_STYLE$C_EXTENDED。

value
OpenVMS usage: integer
type: quadword (unsigned)
access: read
mechanism: by value

プロパティを設定するためのクォドワード値。

prev_value
OpenVMS usage: access_mode
type: quadword (unsigned) address of a quadword value
access: write
mechanism: by reference

プロパティの以前の値を受け取るクォドワードのアドレス。

必要なアクセスまたは特権
なし。

必要な制限値
なし。

関連サービス

$GETJPI

返される条件値

```
SS$NORMAL       The service completed successfully.
SS$ACCVIO       Access violation.
```

B.1.2 $CVT_FILENAME システム・サービス (Alpha システムのみ)

文字列を， RMS 形式からファイル・システム (ACP-QIO) 形式に，またはファイル・システム (ACP-QIO) 形式から RMS 形式に変換します。

形式

SYS$CVT_FILENAME cvttyp ,srcstr ,inflags ,outbuf ,outlen ,outflags

C プロトタイプ

```
int sys$cvt_filename (unsigned int cvttyp,
                      void *srcstr,
                      unsigned int inflags,
                      void *outbuf,
                      unsigned short int *outlen,
                      unsigned int *outflags);
```

引数

cvttyp

```
OpenVMS usage: unsigned_longword
type:          longword (unsigned)
access:        read only
mechanism:     by value
```

RMS 形式から ACP-QIO 形式への変換を行うのか，またはその逆を行うのかを示すロングワード値。

このパラメータで有効な値は，シンボル CVTFNM$C_ACPQIO_TO_RMS および CVTFNM$C_RMS_TO_ACPQIO で表される 2 つです。これらのシンボルは，$CVTFNMDEF マクロによって定義されます。

srcstr

```
OpenVMS usage: string of bytes or words
type:          string of bytes or words
access:        read only
mechanism:     by 32-bit descriptor--fixed length string descriptor
```

このサービスによって変換される文字列。

RMS 形式から ACP-QIO 形式への変換を行う場合，srcstrは ISO Latin-1 文字列またはVTF-7 でエンコードされた文字列です。 ACP-QIO 形式から RMS 形式への変換を行う場合，srcstrはバイト幅またはワード幅の文字から成る文字列です。

記述子長フィールドは，入力文字列がバイト幅とワード幅のどちらであっても，その入力文字列の長さをバイト数で表します。

srcstr引数は，この文字列をポイントする記述子の 32 ビット・アドレスです。

inflags
OpenVMS usage: mask_longword
type: longword (unsigned)
access: read only
mechanism: by value

入力文字列の特性を示すロングワードのフラグ・マスク。

RMS 形式から ACP-QIO 形式に変換する場合， CVTFNM$V_NO_DELIMITERS フラグだけが有効です。

ACP-QIO 形式から RMS 形式に変換する場合，有効なフラグは CVTFNM$V_WORD_CHARS および CVTFNM$V_NO_DELIMITERS ($CVTFNMDEF マクロによって定義される) です。

フラグ	説明
CVTFNM$V_WORD_CHARS	入力ソース文字列には，ワード幅の UCS-2 文字が含まれている (ACPQIO_TO_RMS)。
CVTFNM$V_NO_DELIMITERS	入力ソース文字列は，タイプ区切り文字やバージョン区切り文字としてピリオドやセミコロンが含まれている (または含まれていなければならない) ファイル名として扱うのではなく，任意の文字列 (サブディレクトリなど) として扱わなければならない。

outbuf
OpenVMS usage: string of bytes or words
type: string of bytes or words
access: write only
mechanism: by 32-bit descriptor--fixed-length string descriptor

変換された文字列が書き込まれるバッファ。

RMS 形式から ACP-QIO 形式に変換を行う場合，ソース文字列に含まれている文字によっては，文字列を，バイト幅の ISO Latin-1 文字またはワード幅の UCS-2 文字から構成することができます (ソース文字列内のいずれかの文字で表現するために 1 ワードを必要とする場合には，その出力バッファ内のすべての文字はワード幅になります)。

ACP-QIO 形式から RMS 形式に変換を行う場合，出力文字は， RMS の正規表現の ISO Latin-1 およびVTF-7 文字から構成されています (『Guide to OpenVMS File Applications』を参照してください)。

ACPQIO_TO_RMS 変換の場合，出力文字列がワード幅文字から構成されている場合には，outflagsフラグ・マスク内の CVTFNM$V_WORD_CHARS フラグが設定されます。

outbuf引数は，呼び出し元のモードで書き込み可能なバッファをポイントする 32 ビット・アドレスです。

outlen
OpenVMS usage: word_unsigned
type: word (unsigned)
access: write only
mechanism: by 32-bit reference

outlen引数は，呼び出し元のモードで書き込み可能な (16 ビットの) ワードの 32 ビット・アドレスです。

outflags
OpenVMS usage: mask_longword
type: longword (unsigned)
access: write only
mechanism: by 32-bit reference

出力文字列の特性を示すために，サービスが設定したりクリアするロングワードのフラグ・マスクです。

RMS_TO_ACPQIO 変換で，変換された文字列内の文字が (1 バイト幅ではなく) 1 ワード幅の場合， SYS$CVT_FILENAME は， CVTFNM$V_WORD_CHARS ($CVTFNMDEF マクロによって定義されます) に対応するビットを設定します。変換された文字列内の文字が 1 バイト幅の場合，サービスは CVTFNM$V_WORD_CHARS ビットをクリアします。他のすべてのビットは， RMS_TO_ACPQIO 変換によってクリアされます。

outflags引数は，呼び出し元のモードで書き込み可能な 32 ビットのフラグ・マスクの 32 ビット・アドレスです。

説明

このサービスは，ファイル名 (1) またはサブディレクトリ名 (2) を RMS 形式 (RMS インタフェース上で表示される) と ACP-QIO 形式 (ディスク上に保存される) との間で変換することを目的としています。バージョン 7.2 以前では，これらの形式の表現は同じでしたが，拡張 (ODS-5) ファイル名では，必ずしも同じとは限りません (ODS-5 ファイル指定の詳細については，『Guide to OpenVMS File Applications』を参照してください)。

1. ファイル名は，ファイル名，ファイル・タイプ，およびファイル・バージョンから構成される。
2. サブディレクトリ名は，後ろに ".DIR;1"を追加して，ディスク上の格納するためのディレクトリ・ファイル名を作成する文字列である。

cvttypの値に応じて，このサービスは RMS 形式から ACP-QIO 形式へ，または ACP-QIO 形式から RMS 形式へ，文字列の変換を行います。

ソース文字列はsrcstr引数によって記述され，出力バッファはoutbuf引数によって記述され，結果の文字列の長さはoutlen引数に書き込まれます。

ソース文字列のいずれかが出力バッファのアドレス範囲と重なる場合，出力文字列は予測不可能な値になります。

RMS から ACPQIO への変換:

srcstr記述子引数によって記述される文字列が， ISO Latin-1 または UCS-2 文字列に変換されます。このとき，個々の文字は，$ QIO サービスを経由して ACP-QIO に渡すことができる形式で表現されます。

CVTFNM$V_NO_DELIMITERS 入力フラグがクリアされている場合，ソース文字列はスキャンされ，必要に応じて，$ PARSE が省略時の名前，タイプ，およびバージョン・フィールドの指定なしに実行された場合と同じように，ピリオドおよびセミコロンが挿入または追加されます。スキャンによって名前 (FID なし)，タイプ，またはバージョン以外のフィールドがあることを示す区切り文字を検出した場合，構文エラーが返されます。

CVTFNM$V_NO_DELIMITERS 入力フラグが設定されている場合，個々の文字が評価され，ディスク上の形式に変換されます。ただし，タイプおよびバージョンの区切り文字があるかどうかを判断するためのスキャンは行われず，区切り文字の追加も行われません。

エスケープ文字のサーカンフレックス (^) が前に付いていないパーセント記号 (%) は，疑問符に変換されます。サーカンフレックス (^) が前に付いている ISO Latin-1 文字は，対応する ISO Latin-1 文字に変換されます。サーカンフレックス (^) が前に付いている VTF-7 文字 (^U1234 など) は， UCS-2 文字 (0x1234 など) に変換されます。

いずれかの文字で UCS-2 (ワード幅の文字) による表現が必要な場合，出力文字列中のすべての文字は UCS-2 で表現されます。どの文字でもワード幅の文字による表現が必要でない場合，出力文字列中のすべての文字は ISO Latin-1 (バイト幅) 文字で表現されます。

使用できる文字は， RMS ファイル名 (ファイル名，ファイル・タイプ，ファイル・バージョン) または RMS サブディレクトリ名の中で有効な文字です。たとえば，ディレクトリ区切り文字“[”および“]”は，サーカンフレックス (^) が前に付いている場合を除いて無効です。

ACPQIO から RMS への変換:

srcstr記述子引数によって記述される文字列が，その文字列の RMS 正規表現に変換されます。

CVTFNM$V_NO_DELIMITERS 入力フラグがクリアされている場合，ソース文字列には少なくとも 1 つのセミコロンが含まれていなければならず，さらにそのセミコロンの左側には少なくとも 1 つのピリオドが含まれていなければなりません。このようになっていない場合， RMS$_SYN (構文エラー) が返されます。出力文字列では，これらの 2 つの場合を除くすべてのピリオドおよびセミコロンの前には， RMS のエスケープ文字 (^) が付きます。

CVTFNM$V_NO_DELIMITERS 入力フラグが設定されている場合，出力文字列内のすべてのピリオドまたはセミコロンの前には， RMS のエスケープ文字 (^) が付きます。

inflags引数の CVTFNM$V_WORD_CHARS フラグは，入力文字列にバイト幅 (ISO Latin-1) 文字またはワード幅 (UCS-2) 文字のどちらが含まれているように解釈するかを示します。この引数によってワード幅が含まれていることが示されているにもかかわらず入力の長さの値が奇数になっている場合には，構文エラーが返されます。

疑問符はパーセント記号に変換され，パーセント記号の前にはサーカンフレックス (^) が付きます。 UCS-2 文字は，VTF-7 文字に変換されます。すべての文字は，RMS 正規表現で表現されます。

必要なアクセスまたは特権

なし。

必要な制限値

なし。

関連サービス

なし。

返される条件値

SS$NORMAL	The service completed successfully.
SS$_BADPARAM	Unrecognized conversion type, extraneous input flags set, or zero-length input string.
SS$_INSFARG	Not enough arguments provided.
SS$_TOO_MANY_ARGS	Too many arguments provided.
RMS$_SYN	The service could not translate one or more characters in the strings described by the srcstr argument, the input string has word-width characters but odd byte-length (ACPQIO_TO_RMS only), or the CVTFNM$V_NO_DELIMITERS input flag was clear and the input string did not contain both type and version delimiters.
SS$_BUFFEROVF	The output buffer was not large enough to accommodate the converted string.

B.1.3 $GETJPI システム・サービス

このシステム・サービスには，以下に示す 2 つの新しい項目コードがあります。

JPI$_PARSE_STYLE_PERM
JPI$_PARSE_STYLE_IMAGE

これらの項目コードにより，$SET_PROCESS_PROPERTIES で設定された値 PARSE_STYLE$C_TRADITIONAL または PARSE_STYLE$C_EXTENDED が返されます。戻り値の長さはそれぞれ 1 バイトです。

B.1.4 $CREPRC システム・サービス

stsflgパラメータでは新しいフラグを指定することができます。
PRC$M_PARSE_EXPANDED

これにより， EXPANDED への新しいプロセスに PARSE_STYLE_PERM プロパティおよび PARSE_STYLE_IMAGE プロパティが設定されます。

B.1.5 $SETDDIR システム・サービス

このシステム・サービスについては，次の情報が追加されました。

結果の省略時のディレクトリが 255 バイトを超える場合， Alpha システムでは， Set Default Directory サービスが省略時のディレクトリ文字列を DID に置き換えようとします。この場合，通常の構文チェックに加えて，その指定に対するパス全体が，デバイスを含めてチェックされます。呼び出しが成功するには，このパスが存在していなければなりません。

B.2 レコード管理サービス (RMS) の変更点

OpenVMS のレコード管理サービス (RMS) は， Extended File Specifications をサポートするように変更されています。この後の項では，構文および意味の変更点と，RMS データ構造の変更点について説明します。

B.2.1 レコード管理サービスの変更点の概要

Extended File Specifications をサポートするために，レコード管理サービス (RMS) は，既存のインタフェースを使用して以下のような機能を提供するように強化されました。

- ファイル名，拡張子，およびディレクトリに使用できるより豊富な種類の文字のサポート
- 拡張文字を使用したファイル指定のアクセス
- 8 レベルより深いディレクトリ構造のサポート
- 機能に一定の制約がある条件で， NAM ブロックの使用による 255 バイトより長いファイル指定へのアクセス
- 新しいインタフェース (NAML ブロック) による新しい命名特性を活用した呼び出し元による， 255 バイトより長いファイル指定へのアクセスおよび完全な指定

B.2.1.1 Extended File Specification のサポート

ODS-5 ボリュームでは， RMS が操作できるのは， 8 ビットまたは 16 ビットで 255 文字までの長さのファイル名およびサブディレクトリ指定です。 RMS は，合計で 8 ビットまたは 16 ビットで 512 文字までの長さのパス名を処理できます。

OpenVMS Alpha バージョン 7.2 以前は， NAM ブロック・インタフェースが渡すことのできるファイル指定は，それぞれ (結果のファイル指定の場合も) 最大で 255 バイトまででした。この後の項では，より長いファイル指定を渡すための変更点と，このリリース以前の NAM ブロック・インタフェースを使用したアプリケーションに対して提供する互換性について説明します。

B.2.1.2 追加された文字

ODS-5 ボリューム上では， RMS は，任意の数の 8 ビット文字が含まれている名前を持つファイルおよびディレクトリへのアクセスをサポートしています。ただし， C0 制御セット (16 進数で 00 〜 1F) および以下の文字は例外です。

二重引用符 (")
アスタリスク (*)
バックスラッシュ (\ )
Colon (:)
左山括弧および右山括弧 (< >)
スラッシュ (/)
疑問符 (?)
縦線 (|)

この中には， C1 文字セット(16 進数の 80 〜 9F) の他， 9F 〜 FF までのグラフィック文字およびその他の文字も明示的に含まれることに注意してください。これにより，すべての ISO Latin-1 文字セットおよび定義済みの Unicode 文字を使用することができます (すでに示されているとおり 7 ビット文字は例外です)。

B.2.1.3 深くネストされたディレクトリのサポート

Alpha システム上の Extended File Specifications では， RMS は最大で 255 レベルまでのディレクトリの深いネストをサポートしています。ただし，合計のディレクトリ指定が 8 ビットまたは 16 ビットで 512 文字を超えてはならないという制約があります。ディレクトリの深いネストは， ODS-2 ディスクでもサポートされています。

B.2.2 構文および意味の変更点

この後の項では，新しい RMS のファイル指定構文および意味の機能について説明します。 Extended File Specifications の環境で RMS を使用する方法の詳細については，『Guide to OpenVMS File Applications』を参照してください。

B.2.2.1 ファイル名の最初の文字としてのハイフンの使用

OpenVMS バージョン 7.2 以前の RMS ドキュメントでは，ハイフン (マイナス記号) で始まるファイル名を作成しないように推奨されていました。

Alpha システムでは， Extended File Specifications により RMS に変更が追加され，ファイル名またはディレクトリ名のどの位置にもハイフンを使用できるようになりました。ハイフンが含まれているディレクトリ名があいまいである，つまり，親ディレクトリを参照するように解釈される可能性がある場合，そのハイフンにエスケープ文字のサーカンフレックス (^) を付けてファイルまたはディレクトリを正しく指定しなければなりません。

B.2.2.2 直接受け付けられる文字

ファイル指定の中で (特殊なエスケープ文字なしで) RMS インタフェースから使用できる文字セットは，次のリストのように拡張されています。これらの文字の前には，エスケープ文字のサーカンフレックス (^) を使用できないことに注意してください。

- 大文字と小文字の英数字

  A - Z, a - z, 0 - 9

- 特殊なASCII (7 ビット) 文字

  $ - ‘ _ ~

- 16 進数で A0 ～ FF までの範囲の ISO Latin-1 文字

B.2.2.3 エスケープ文字を必要とする文字

- エスケープ文字 (^) の後に 16 進数が続く場合は，その後に 2 番目の 16 進数が必要である。後に続く 2 文字は，任意の 8 ビット文字の 16 進数値を表す。

  たとえば，^20 はスペースを表す。

- エスケープ文字の後にアンダスコア (^_) またはスペースが続くと， 1 つのスペースを表す。

- エスケープ文字の後に大文字の U (^U) が続くと，この後の 4 文字が任意の 16 ビット文字の 16 進数値を表すことを示す。

- 2 バイトの Unicode 文字はそれぞれ^Uxxxx というシーケンスとして表現しなければならない。

- 以下の文字を， RMS および DCL への入力でファイル名の一部として使用するときには，エスケープ文字が必要になる。

  感嘆符 (!)
  ポンド記号 (#)

アンパサンド (&)
一重引用符 (')
左括弧 (( )
右括弧( ))
正符号 (+)
アットマーク (@)
左中括弧 ({)
右中括弧 (})
ピリオド (.)[1]
コンマ (,)
低アクセント (`)
セミコロン (;)
左大括弧 ([)
右大括弧 (])
パーセント記号 (%)
サーカンフレックス (^)
等号 (=)

B.2.2.4 エスケープ文字を付けることができる文字

以下の文字の前には， RMS または DCL への入力の際にエスケープ文字 (^) を付けることができますが，必須ではありません。

ドル記号 ($)
負記号 (-)
チルダ (~)[2]
ピリオド (.)[1]

B.2.2.5 予約済みのエスケープ・シーケンス

これまでの項で説明されていない文字が後に続くエスケープ文字から成るシーケンスは，予約されています。

B.2.2.6 ファイル指定の正規表現

同じ文字を表現する方法が複数存在することがあります。たとえば，^20，^，および^_はすべて等価です。 RMS がファイル指定を (結果の名前などとして) 出力するときには，以下の規則に従って使用する規則を決定します。

- 8 ビットで表現できない文字は，^Uxxxx として表現される。このとき， xxxx は 4 桁の 16 進数である。
- スペースは，エスケープ文字の後にアンダスコアを続けて表現される (^_)。

[1] エスケープ文字は，ディレクトリ名の中のピリオドの前には必須ですが，ファイル名の中のピリオドの前には付けても付けなくても構いません。ただし，ファイル・タイプを区切るピリオドとしては使用できません。ピリオドは，ファイル・タイプの中で使用できません。

[2] ファイル名またはディレクトリの中で先頭の文字としてチルダを使用する場合，エスケープ文字が必要なことがあります。「◆チルダ (~) をファイル名の最初の文字として使用しない」を参照してください。

- グラフィック表現を持っていないか，あるいは他の OpenVMS ソフトウェアまたはターミナルによって制御機能を実行するために使用される，その他の 8 ビットの ISO Latin-1 文字は，エスケープ文字の後に 2 桁の 16 進数を続けて表現される (^xx)。それ以外の場合，そのような文字は文字そのものを表す。以下の 8 ビットの値は，エスケープ文字の後に 2 桁の 16 進数を続けて出力される。

  7F (rubout)
  80 〜 9F (C1 制御文字)
  A0 (区切りでないスペース)
  FF (ラテン語の小文字 y による分音記号)

- ファイル指定の長さが 255 バイトを超え， NAM ブロックを使用して出力しなければならない場合には， DID または FID による短縮が使用される。
- 以下の文字は，前にサーカンフレックス (^) を付けて出力される。

  感嘆符 (!)
  ポンド記号 (#)
  アンパサンド (&)
  アポストロフィ (')
  低アクセント (‘)
  左括弧 (( )
  右括弧( ))
  正符号 (+)
  アットマーク (@)
  左中括弧 ({)
  右中括弧 (})
  ピリオド (.)
  コンマ (,)
  セミコロン (;)
  左大括弧 ([)
  右大括弧 (])
  パーセント記号 (%)
  サーカンフレックス (^)
  等号 (=)

B.2.2.7 DID による短縮

拡張ファイル名を使用すると，長いディレクトリのレベルが多すぎたり，長いファイル名にエスケープ文字が含まれていることが原因で，有効な名前であっても変更されていない RMS アプリケーションや DCL にとっては長すぎて処理できない場合があります。従来の (またはバージョン 7.2 以前の) インタフェースを使用しているアプリケーションとの互換性を保つために， RMS がアプリケーションに出力したり，アプリケーションが従来のインタフェースを使用して RMS に入力できる短い名前が生成されます。

ファイル指定の長さが 255 バイトを超える場合には，RMS はまず DID による短縮を使用して名前を作成します。RMS は，ディレクトリをそのディレクトリ ID (DID) に短縮することによって，適切な短い名前を生成しようとします。次に例を示します。

```
DKA100:[5953,9,0]FOO.TXT;1.
```

DID による短縮の制約

従来の (バージョン 7.2 以前の) 短い出力バッファに納まらないファイル指定を処理するときに，RMS はルートまたはディレクトリのコンポーネントを，そのコンポーネント内の最も下のレベルのサブディレクトリのディレクトリ ID に置き換えて短縮しようとすることがあります。

RMS が DID によって短縮されたルートまたはディレクトリのコンポーネントを生成できない場合もあります。たとえば，最も下のレベルのサブディレクトリにワイルドカードが含まれている場合には，RMS が使用できる特定のディレクトリ ID はありません (RMS はルートまたはディレクトリのコンポーネントの一部を DID に置き換えることもしません)。

RMS が DID によって短縮されたルートまたはディレクトリを生成することによって，ファイル指定を適切に短縮できない場合には，次のステップである FID による短縮に進みます。

B.2.2.8 FID による短縮

DID による短縮を実行してもまだファイル指定が長すぎる場合，RMS は次に，ファイル名フィールドの中でファイルをそのファイル ID (コンマで区切られた 10 進数の文字列のシーケンスを大括弧で囲んだもの) に短縮することによって，適切な短い名前を生成しようとします。

拡張フィールドが通常どおり表示されている場合，スペースに余裕があるかどうかによって，生成された名前には完全な拡張子が含まれるか，または拡張子が (ピリオド (.) も共に) 省略されます。

バージョン番号が通常どおり表示されている場合，FID によって短縮されたファイル名にはバージョン番号が含まれます。ファイルを認識しやすくするために，FID による短縮が生成されると，名前フィールドには実際のファイル名の先頭部分のサブセットも含まれます。このサブセットには，ファイル名の最初の (エスケープ文字も文字として認識される) 38 文字と，後に続くチルダ (~) から構成されます。ファイル名の最初の 38 文字以降だけが異なるファイルがある場合でも，ファイルの曖昧さは解決されません。

次に FID による短縮の例を示します。

```
LookAtWhatWeHave^!ThisIsAVery_long^.fi~[7254,30,0].txt;1
```

FID および DID による短縮の制約

FID によって短縮されたファイル名は，RMS への入力に使用できますが，RMS への入力の際に有意義なのは FID による短縮だけです。この時に使用するサブセットのファイル名，タイプ・フィールド，およびバージョン番号はすべて，入力の際には無視されます。

ファイルを作成するときに，FID を CREATE コマンド (または他のファイル作成コマンドまたは API) への入力として使用することはできません。DID を使用してディレクトリを作成することはできません。ファイル指定に FID を使用すると，そのファイル指定には最初からデバイスおよびディレクトリが含まれているか，または RMS で処理される省略時の値からデバイスおよびディレクトリを取得します。指定された FID を持つファイルが既にボリューム上に存在する場合には，指定されたディレクトリ内に存在していなければなりません。そうでない場合，RMS はあたかもそのファイルが見つからなかったかのように動作します。

B.2.3 RMS のデータ構造の変更点 (Alpha システムのみ)

この項では，名前 (NAM) ブロックの変更点を示します。また，長さが 255 バイトを超えるファイル指定を指定するために使用する新しい名前 (NAML) ブロックについても説明します。

B.2.3.1 NAM ブロック

ファイル名ブロック・オプション・フィールド NAM$B_NOP には，以下の新しいフラグがあります。

フラグ	意味
NAM$V_NO_SHORT_UPCASE	ユーザによって設定され，NAM$L_ESA バッファにあるディレクトリおよびファイル指定を大文字に変換しないように RMS に指示する。

ファイル名状態ビット・フィールド NAM$L_FNB には，以下の新しいフラグがあります。

フラグ	意味
NAM$V_DIR_LVLS_G7	ディレクトリ・レベルの数が 7 を超えることを示す。このフラグが設定されている場合，NAM$V_DIR_LVLS は 7 に設定される。
NAM$V_WILD_SFDG7	7 を超えるレベルにあるサブディレクトリに，ワイルドカード文字が含まれていることを示す。このフィールド・オフセットは，NAM$V_WILDCARD フィールド・オフセットに要約される。

NAM には，NAM$B_NMC，NAM$W_FIRST_WILD_DIR，および NAM$W_LONG_DIR_LEVELS という 3 つのフィールドがあります。NAM$B_NMC フィールドは，以下のフラグを返します。

フラグ	意味
NAM$V_DID	入力ディレクトリのルートまたはディレクトリ名のコンポーネントの中に DID によって短縮されたディレクトリが見つかると， RMS によって設定される。
NAM$V_FID	入力ファイル指定の中に FID によって短縮されたファイル名が見つかると， RMS によって設定される。
NAM$V_RES_DID	短い結果または拡張バッファの中に DID によって短縮されたディレクトリがあると， RMS によって設定される。
NAM$V_RES_FID	短い結果または拡張バッファの中に FID によって短縮された名前があると， RMS によって設定される。
NAM$V_RES_ESCAPE	短い結果または拡張バッファの中にエスケープ文字 (^) があると， RMS によって設定される。
NAM$V_RES_UNICODE	短い結果バッファまたは拡張バッファの中に 1 つまたは複数の^U シーケンスがあると， RMS によって設定される。

B.2.3.2 NAML ブロック

NAML は，オプションで NAM ブロックの代わりに使用できる新しいブロックです。NAML には， NAM のすべてのフィールドの他に， 255 バイトを超える長さのファイル指定を指定できるようにするための新しいフィールドが含まれています。

以下に NAML の新しいフィールドを示します。

拡張フィールド名	サイズ (バイト)	意味
NAML$L_FILESYS_NAME	4	ユーザが指定するファイル・システム名バッファのアドレス。 RMS が NAML$V_FILESYS_NAME_UCS2 出力フラグを設定すると，出力ファイル名は 2 バイト文字になり，ファイル名の中の ASCII 文字および区切り文字を含むすべての文字が 2 バイト文字になる。それ以外の場合は，すべて 1 バイト文字になる。
NAML$L_FILESYS_NAME_ALLOC	4	ユーザが指定するファイル・システム名バッファの割り当てサイズ。
NAML$L_FILESYS_NAME_SIZE	4	RMS によって返されるファイル・システム名の長さ。
NAML$L_LONG_DEFNAME_SIZE	4	入力として指定される省略時のファイル指定文字列の長さ (long)。 FAB$B_DNS と等価 (入力のみ)。 FAB$L_DNA が -1 に設定され， FAB$B_DNS が 0 に設定されている場合に限って使用される。
NAML$L_LONG_DEFNAME	4	入力として指定される省略時のファイル指定文字列のアドレス (long)。 FAB$L_DNA と等価 (入力のみ)。 FAB$L_DNA が -1 に設定され， FAB$B_DNS が 0 に設定されている場合に限って使用される。
NAML$L_LONG_FILENAME_SIZE	4	ファイル指定文字列のサイズ (long)。 FAB$B_FNS と等価 (入力のみ)。 FAB$L_FNA が -1 に設定され， FAB$B_FNS が 0 に設定されている場合に限って使用される。
NAML$L_LONG_FILENAME	4	ファイル指定文字列のアドレス (long)。 FAB$L_FNA と等価 (入力のみ)。 FAB$L_FNA が -1 に設定され， FAB$B_FNS が 0 に設定されている場合に限って使用される。
NAML$L_LONG_NODE_SIZE	4	ノード名文字列の長さ (long)。
NAML$L_LONG_NODE	4	ノード名文字列のアドレス (long)。
NAML$L_LONG_DEV_SIZE	4	デバイス文字列の長さ (long)。

拡張フィールド名	サイズ (バイト)	意味
NAML$L_LONG_DEV	4	デバイス文字列のアドレス (long)。
NAML$L_LONG_DIR_SIZE	4	ディレクトリ文字列の長さ (long)。
NAML$L_LONG_DIR	4	ディレクトリ文字列のアドレス (long)。
NAML$L_LONG_NAME_SIZE	4	ファイル名文字列の長さ (long)。
NAML$L_LONG_NAME	4	ファイル名文字列のアドレス (long)。
NAML$L_LONG_TYPE_SIZE	4	ファイル・タイプ文字列の長さ (long)。
NAML$L_LONG_TYPE	4	ファイル・タイプ文字列のアドレス (long)。
NAML$L_LONG_VER_SIZE	4	ファイル・バージョン文字列の長さ (long)。
NAML$L_LONG_VER	4	ファイル・バージョン文字列のアドレス (long)。
NAML$L_LONG_EXPAND_ALLOC	4	拡張文字列領域のサイズ (long)。拡張バッファのサイズ (long) を指定するために，呼び出し元によって設定される。
NAML$L_LONG_EXPAND_SIZE	4	拡張文字列の長さ (long)。拡張文字列の長さ (long) を表すために， RMS によって設定される。
NAML$L_LONG_EXPAND	4	拡張文字列領域のアドレス (long)。拡張バッファ (long) をポイントするために，呼び出し元によって設定される。
NAML$L_LONG_RESULT_ALLOC	4	結果文字列領域のサイズ (long)。結果バッファのサイズ (long) を指定するために，呼び出し元によって設定される。
NAML$L_LONG_RESULT_SIZE	4	結果文字列の長さ (long)。結果文字列の長さ (long) を表すために， RMS によって設定される。
NAML$L_LONG RESULT	4	結果文字列領域のアドレス (long)。結果バッファ (long) をポイントするために，呼び出し元によって設定される。
NAML$L_INPUT_FLAGS	4	この後の表で定義されている NAML$V_NO_SHORT_OUTPUT など， RMS への入力として指定されるその他のフラグ。
NAML$L_OUTPUT_FLAGS	4	この後の表で定義されている NAML$V_LONG_RESULT_ESCAPE および NAML$V_FILESYS_NAME_UCS2 など， RMS による出力として渡されるその他の状態ビット。
NAML$Q_USER_CONTEXT	8	呼び出し元は，このフィールドをどのような目的にでも使用できる。 RMS によって読み込まれたり変更されることはない。

RMS は， NAML$L_INPUT_FLAGS フィールドから以下のフラグを読み込みます。

フラグ	意味
NAML$V_NO_SHORT_OUTPUT	RMS が NAM$L_ESA または NAM$L_RSA バッファに入力を行わないようにするために，ユーザによって設定される。

RMS は， NAML$L_OUTPUT_FLAGS フィールドに以下のフラグを書き込みます。

フラグ	意味
NAML$V_FILESYS_NAME_UCS2	NAML$L_FILESYS_NAME によってポイントされる名前が 2 バイトの Unicode 文字 6 個から構成されている場合に，RMS によって設定される。
NAML$V_LONG_RESULT_DID	long の結果バッファまたは拡張バッファの中に DID によって短縮されたディレクトリがある場合に，RMS によって設定される。
NAML$V_LONG_RESULT_ESCAPE	long の結果バッファまたは拡張バッファの中にエスケープ文字 (^) がある場合に，RMS によって設定される。
NAML$V_LONG_RESULT_FID	long の結果バッファまたは拡張バッファの中に FID によって短縮された名前がある場合に，RMS によって設定される。
NAML$V_LONG_RESULT_UNICODE	long の結果バッファまたは拡張バッファの中に，1 つまたは複数の^U シーケンスがある場合に，RMS によって設定される。

B.2.3.2.1 NAML ブロックの有効性の確認

RMS に渡された名前 (FAB$L_NAM を参照) に NAML$C_BID に等しいブロック識別子 (NAML$B_BID を参照) が含まれている場合，RMS は以下の有効性チェックを行います。

1. NAML$B_BLN フィールドが NAML$C_BLN と正確に一致している。
2. NAML$L_LONG_RESULT_ALLOC および NAML$L_LONG_EXPAND_ALLOC の合計が，NAML$C_MAXRSS 以下である。
3. すべての未使用フィールド (MBZ が含まれたシンボリック名を持つ) に 0 (ゼロ) が含まれている。フィールドを初期化する前に全体の構造をクリアすると，この条件を満たすことができる。

これらの有効性チェックのうちいずれかが失敗すると，RMS$_NAML エラー状態が返されます。

B.2.3.2.2 NAM および NAML ブロックの使用

NAML には，すべての NAM フィールドと等価なフィールドに加えて，より長いファイル指定に対応するために 28 個の新しいフィールドが含まれています。NAML フィールドには FDL 属性はありません。

NAML の新しいフィールドの多くは NAM フィールドに対応していますが，より長い名前を使用できるフィールドです。たとえば，NAML$L_LONG_EXPAND，NAML$L_LONG_EXPAND_ALLOC，および NAML$L_LONG_EXPAND_SIZE というフィールドは，NAM$L_ESA，NAM$B_ESS，および NAM$B_ESL に対応していますが，255 バイトを超えるより長い名前を使用できます。このような対応関係にあるフィールドがある場合には，元のフィールドは「短いフィールド」と呼ばれ，対応するフィールドは「長いフィールド」と呼ばれます。

RMS が，長いバージョンと短いバージョンの両方を持つ NAML のフィールドに情報を書き込むときには，RMS は通常，等しい情報を両方のフィールドに書き込みます。短いフィールドまたは長いフィールドのいずれか一方が小さすぎて情報を格納できない場合には，RMS はまず短いフィールドに納まるようにファイル指定を短縮し

ようとしますが，エラーを返します。 RMS が短いフィールドに書き込まないように指示するフラグNAML$V_NO_SHORT_OUTPUTを指定すると，短いフィールドでのこのようなエラーの発生を防ぐことができます。ただし， NAML を使用している場合には， RMS は常に長いフィールドを使用します。 RMS が長いフィールドを使用しないようにするには， NAML ではなく NAM を使用しなければなりません。

RMS が，長いバージョンと短いバージョンの両方を持つ NAML のフィールドから情報を読み込むときには， RMS は通常，長いフィールドから読み込みます。 RMS が短いフィールドから読み込むようにするには， NAML ではなく NAM を使用します。 RMS がこれらのフィールドから読み込むもっとも一般的なタイミングは，PARSE の後の$SEARCH の実行中で， NAML の場合は NAML$L_LONG_EXPAND によってポイントされているバッファ， NAM の場合は NAM$L_ESA によってポイントされているバッファから RMS が読み込みを実行するときです。さらに，NAM または NAML が関連する名前ブロックとして使用されている場合， RMS は，NAML の場合は NAML$L_LONG_RESULT によってポイントされているバッファ， NAM の場合は NAM$L_RSA によってポイントされているバッファから情報を読み込みます。

RMS が NAM および NAML を処理する方法にはこのような違いがあるため，NAML に接触する可能性のあるコードは，それが NAM ではなく NAML であることを認識できることが重要です。 NAM でルーチンが実行する可能性のある処理の中には， NAML では期待どおりに動作しないものがあります。たとえば，あるルーチンが NAML のコピーを作成する一方でコピーする長さとして NAM$C_BLN 定数を使用した場合，結果は無効な NAML になります。また，別のあるルーチンが，呼び出し元のルーチンに影響を与えることなく NAM を使用できると想定して NAM$L_ESA および NAM$L_RSA によってポイントされたバッファを置き換えると， NAML$L_LONG_EXPAND および NAML$L_LONG_RESULT によってポイントされたバッファが失われます。

このため， OpenVMS で提供される API が以前のバージョンの NAM を (直接，または FAB を使用して間接的に) 返していた場合，その API は，呼び出し元による明示的なアクション (通常はフラグ・ビットを設定すること) がない限り， NAML を返さないという規則に従うことになっています。他の API も同じ規則を使用することを推奨します。さらに， NAML を認識するアプリケーションが NAML を API に渡す場合には，その API が NAM セクションだけを使用する (たとえば， NAML$V_NO_SHORT_OUTPUT ビットを設定しない) ように注意しなければなりません。

NAM または NAML のいずれかを受け付けるルーチンを作成している場合には，NAM または NAML のどちらがあるのかを判断する必要があります。 NAML があり， RMS が NAML の中に残した情報を読み込む場合には，長いフィールドの情報を参照します。さらに， NAM ブロックまたは NAML ブロックを他の場所にコピーする場合は，十分注意して構造自体に格納されている長さを使用してコピーできる量を判断しなければなりません。 NAM$B_BLN にはその構造の実際の長さが含まれているため， NAM$C_BLN 定数ではなく，実際にコピーしている構造にある

NAM$B_BLN フィールドを使用する必要があります。 NAM の長さであるシンボル NAM$C_BLN を使用すると， NAML の場合には短かすぎます。

B.2.3.2.3 **返される条件値** 表 B–1 は， NAML ブロックを使用するときにそれぞれの RMS サービスに返される条件値を示しています。

表 B–1 NAML ブロックを使用したときに返される RMS 条件値

RMS サービス	返される条件値
$CREATE	RMS$_NAML RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$DISPLAY	RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ
$ENTER	RMS$_NAML RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$ERASE	RMS$_NAML RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$OPEN	RMS$_NAML RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$PARSE	RMS$_NAML RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ
$REMOVE	RMS$_NAML RMS$_NALFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$RENAME	RMS$_NAML RMS$_NAMLESS RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS
$SEARCH	RMS$_NAML RMS$_NAMLFSINV RMS$_NAMLFSSIZ RMS$_NAMLRSS

B.3 Files–11 XQP の変更点

注意

この項に含まれているファイル・システムに関する情報は，現在このドキュメントでのみ扱われています。

Files–11 Extended QIO Processor (XQP) ファイル・システムは，$QIO インタフェースを使用して拡張ファイル名をサポートするように強化されました。 XQP ファイル形式の規則は， RMS によって提供されているような，ファイル名を受け付ける他のシステム・サービスに適用される規則とは異なる場合があることに注意してください。 RMS で使用される新しい構文およびセマンティックの詳細については，第 B.2.2 項を参照してください。

機能強化された XQP では，以下の Extended File Specifications 機能をサポートしています。

- 8 ビットの ISO Latin-1 文字セットおよび 16 ビットの Unicode (UCS-2) 文字セットなどの，より豊富な文字の使用
- 長いファイル名
- ファイル名での大文字と小文字の区別の保存 (最初に作成されたとおりに保存する)

第 B.3 節のこの後の項では，Files–11 XQP ファイル・システムおよび$QIO インタフェースに追加された変更点についてより詳しく説明します。

B.3.1 ファイルの命名および形式の変更点

OpenVMS バージョン 7.2 以前では，Files–11 XQP でサポートされる有効なファイル名は，ファイル名およびファイル・タイプが共に 39 文字以内で，合計で 85 個の ASCII 文字[2]に制限されていました。拡張ファイル名をサポートするために，このような制約が緩和され，以下の内容がサポートされるようになりました。

- ファイル名での 8 ビットの ISO Latin-1 Multinational 文字セット (ASCII はこのサブセット) のほとんどの文字の使用 (以下の例外を除く)

  C0 制御セット (16 進数の 00 ～ 1F)
  左山括弧 (<)
  右山括弧 (>)
  コロン (:)
  スラッシュ (/)
  バックスラッシュ (\ )

[2] ファイル名の 39 文字，区切り文字の 1 文字 (.)，ファイル・タイプの 39 文字，区切り文字の 1 文字 (;)，バージョン番号の 5 文字を合わせて 85 文字。

縦線 (|)
疑問符 (?)
アスタリスク (*)

C1 文字セット (16 進数の 80 ～ 9F) の他，9F ～ FF のグラフィック文字および他の文字も明示的に**含まれている**ことに注意。

- ファイル名の中のピリオド (.)
- 16 ビットの Unicode 文字 (UCS-2) を使用してエンコードされたファイル指定およびディレクトリ指定
- 区切り文字の 1 文字 (.) を含み，8 ビットまたは 16 ビット文字で合計 236 文字以内の，より長いファイル名およびファイル・タイプ
- 8 ビットの 242 文字まで[3]，または 16 ビットで 124 文字まで[4]のファイル指定
- 大文字と小文字が混在したファイル指定や，小文字だけのファイル指定が，大文字に変換されることなくディスクに格納される機能 (大文字と小文字の区別の保存)

これらの変更点は，Files–11 ODS-5 形式に初期化または変換されたボリュームに関してのみ適用されます。現在 ODS-2 ボリュームで使用されているセマンティックおよび動作に依存しているアプリケーションは，引き続きこれまでと同様に機能します。

B.3.1.1 入力ファイル名の形式の指定

ファイル指定は，QIO P2 パラメータを使用し，ディスクリプタによってファイル・システムに渡されます。ディスクリプタには，ファイル指定のテキストへのポインタと，ファイル指定の長さの合計をバイト単位で表した長さのフィールドが含まれています。

ファイル指定の形式は，新しい FIB$B_NAME_FORMAT_IN フィールドで識別することができます。この形式は，表 B–2 で示されている値のいずれかを取ります。

表 B–2 ファイル形式の FIB 定数

形式の値	形式のタイプ
FIB$C_ODS2	ODS-2 形式
FIB$C_ISO_LATIN	ISO Latin-1 形式
FIB$C_UCS2	Unicode (UCS-2) 形式

指定された形式がファイル・システムで認識される形式でない場合には，SS$_BADPARAM エラーが返されます。それ以外の場合には，ファイル・システムは，指定された形式で定義された規則に従って，ファイル指定を解析しようとしま

[3] ファイル名の 236 文字，区切り文字 (.)，ファイル・タイプ，区切り文字の 1 文字 (;)，バージョン番号の 5 文字を合わせて 242 文字。
[4] ファイル名の 118 文字，区切り文字 (.)，ファイル・タイプ，区切り文字の 1 文字 (;)，バージョン番号の 5 文字を合わせて 124 文字。

す。ファイル名の解析に失敗すると， SS$_BADFILENAME エラーまたは SS$_BADFILEVER エラーが返されます。

ファイル・システムに渡された FIB に FIB$B_NAME_FORMAT_IN フィールドが含まれていない場合，ファイル・システムは，指定されたファイル指定が ODS-2 形式であると想定します。これは，変更されていないプログラムとの互換性を保つために行われます。

ファイル・システムは，ファイル指定をボリュームに格納する前に，ファイル指定を最も単純な互換形式に変換します。たとえば， 0x00FF より大きい文字の値が含まれていないUnicode (UCS-2) 形式で指定されたファイル指定は，ボリュームに格納される前に， ISO Latin-1 形式に変換されます。

B.3.1.2 返されるファイル名の形式の制御

ファイル・システムは，ファイル指定を返すときにファイル形式を新しい FIB$B_NAME_FORMAT_OUT フィールドに書き込みます。表 B–2 に示されているいずれかの値が使用されます。

ただし，すべてのプログラムがすべての利用可能な命名形式を処理できるとは限りません。 QIO システム・サービスの呼び出し元が，表 B–3 に示されている新しい FIB$W_NMCTL フラグを使用して，返される形式を選択することができます。

表 B–3 新しい FIB$W_NMCTL フラグ

フラグ名	解釈
FIB$V_NAMES_8BIT	呼び出し元は (8 ビットの) ODS-2 形式および ISO Latin-1 形式を受け付けることができる。
FIB$V_NAMES_16BIT	呼び出し元は (16 ビットの) Unicode (UCS-2) 形式を受け付けることができる。

これらの新しいフラグは，返されるファイル指定を以下のように制御します。

- 両方のフラグがクリアされている場合

  ODS-2 形式の名前だけが返される。これには元は ISO Latin-1 形式または Unicode (UCS-2) 形式だったものが，ボリュームに格納される前に ODS-2 に変換されたものも含まれることに注意が必要である。すべての指定は，返される前に大文字に変換される。

- FIB$V_ NAMES_8BIT が設定され
  FIB$V_ NAMES_16BIT がクリアされている場合

  ODS-2 形式および ISO Latin-1 形式で格納されているファイル指定だけが返される。 FIB$B_NAME_FORMAT_OUT フィールドの値は，返される特定の名前の形式を示している。 ODS-2 形式のファイル指定は，返される前に大文字に変換されない。

- FIB$V_ NAMES_8BIT がクリアされ
  FIB$V_ NAMES_16BIT が設定されている場合

すべてのファイル指定は Unicode (UCS-2) 形式で返される。

- 両方のフラグが設定されている場合

  ファイル指定はボリュームに格納されているとおりの形式で返される。これは，ファイル名構文とそこに含まれる文字との互換性を保つ最も単純な形式である。たとえば，元は Unicode 形式で， ISO Latin-1 文字セットの一部である文字だけが含まれているファイル指定は， ISO Latin-1 形式で返される。

B.3.1.3 ワイルドカードの検索と疑似名

ファイル・システム操作によって返されるファイル指定は，通常は呼び出し元のプログラムが理解できる形式になっています。しかし，入力指定にワイルドカードが含まれている操作の場合にはこのようになりません。たとえば，以下の ODS-2 準拠のファイル指定の中にあるワイルドカードが，この場合に相当します。

```
A*.DOC
```

これは， ISO Latin-1 ファイル指定に対応している可能性があります。

```
A sample name with periods.and.other;punctuation#in the name.doc;1
```

返されるファイル指定には区切り文字としてのピリオドが 1 つしか含まれていないと想定するアプリケーションは，正しく動作しない可能性があります。ファイル・システムは，呼び出しもとのプログラムがエラーになるようなファイル指定を返すのではなく，代わりに疑似名を返します。実際に返される疑似名は，以下の表に示すように，疑似名が表現する名前のタイプによって異なります。

ファイル形式	疑似名の例
ISO Latin-1 (FIB$C_ISL1)	\ pISO_LATIN\ .???
Unicode (FIB$C_UCS2)	\ pUNICODE\ .???

ファイル・システムは，呼び出し元のプログラムがどの形式を理解できるかを，FIB$V_NAMES_8BIT および FIB$V_NAMES_16BIT フラグの設定から判断します。これらのフラグは，表 B–4 で示すように，返される名前の形式を制御します。

表 B–4 FIB フラグの設定とそれに従って返される名前の形式

FIB フラグの設定		ファイル形式		
8 ビット	16 ビット	ODS-2	ISO Latin-1	Unicode
false	false	ODS-2	疑似名	疑似名
true	false	ODS-2	ISO Latin-1	疑似名
false	true	UCS-2	UCS-2	UCS-2
true	true	ODS-2	ISO Latin-1	UCS-2

ファイル・システムが疑似名を返すとき，ユーザまたは呼び出し元のアプリケーションに，直接のファイル・アクセスを許可せずに疑似名を返す対象となるファ

イルについて通知します。このため，疑似名には，入力ファイル指定としては**有効でない**文字が含まれています。疑似名を使用してファイルを操作しようとすると，SYSTEM-F-BADFILENAME エラーが返されます。

バッファ・サイズ

表 B–5 は，すべての返される可能性のあるファイル指定を格納するために，各バッファで必要な最小サイズを示しています。

表 B–5 各ファイルの安全なバッファ・サイズ (バイト単位)

ファイル形式	QIO の最小値	XQP の最小値
ODS-2	86	86
ISO Latin-1	264	243
Unicode	538	486

個々のアプリケーションでの上限は，そのアプリケーションがサポートしている形式によって異なります。 XQP の最小値は， QIO インタフェースを使用する他のファイル・システムの通常の最小値よりも小さいことに注意してください。これは， XQP によって課せられる，ファイル指定に関する 236 バイトのサイズ制限のためです。

ファイル指定が用意されたバッファよりも長い場合，ファイル・システムは，エラーを生成せずに返されたファイル指定の長さを切り捨てます。ファイル指定が用意されたバッファよりも短い場合，ファイル指定の最後からバッファの最後までの間の空きスペースには 0 (ゼロ) が埋め込まれます。

B.3.1.4 変更されていないアプリケーションとの互換性

インタフェースの新しい機能を利用するように変更されていないアプリケーションは，省略時の設定では，以下の条件のいずれか一方が満たされていれば， ODS-2 準拠のファイル指定または疑似名だけを受け取ります。

- FIB$V_NAMES_8BIT フラグおよび_16BIT フラグをクリアしたままにしている。
- 新しい FIB$B_NAME_FORMAT_IN フィールドおよび FIB$B_NAME_FORMAT_OUT フィールドが含まれていない FIB を指定している。

小文字だけが含まれているファイル指定は， ODS-2 では有効なはずですが，返される前に大文字に変換されます。

変更されていないアプリケーションによって入力パラメータとして指定されたファイル指定は， 8 ビットの ODS-2 対応の名前として解釈されます。この名前の有効性は，既存の ODS-2 解析規則を使用してチェックされます。 ODS-5 ボリューム上では，このファイル指定はディスクに保存される前に大文字に変換されません。前に説明したような変換が実行されるため，変更されていないアプリケーションは，このファイル指定を大文字によるファイル指定として認識するためです。ただし，新しい FIB フラグのうちいずれかを設定したアプリケーションは，このファイル指定を大文字と小文字が混在したファイル指定として認識します。

B.3.2 ファイル属性の変更点

この後の項では，ODS-5 で導入された新しいファイル属性と，既存の属性のセマンティックの変更点について説明します。

B.3.2.1 変更されたファイル属性

表 B–6 は，ODS-5 ボリューム上のファイルについて，変更または制約された属性を示しています。

表 B–6 変更された属性コード

属性名	最大サイズ (バイト単位)	意味
ATR$C_ASCNAME	252	ファイル・ヘッダに格納されているファイル指定
ATR$C_FILE_SPEC	4098	デバイス，最適トライ・パス，およびファイル指定
ATR$C_FILNAM	10	Radix–50のファイル名
ATR$C_FILTYP	4	Radix–50のファイル・タイプ
ATR$C_FILVER	2	Radix–50のファイル・バージョン

ATR$C_ASCNAME

ATR$C_ASCNAME 属性を使用すると，ファイルの 1 次ヘッダに格納されているファイル指定の読み書きができるようになります。

ATR$C_ASCNAME **属性の読み込み**

ODS-2 ボリュームでは， ASCNAME 属性は以前と同様に返されます。 ODS-5 ボリュームでは，ファイル指定は用意されたバッファに返され，名前の形式は新しい FIB$B_ASCNAME_FORMAT セルに返されます。

名前が返される形式は，出力ファイル指定のパラメータと同様に， FIB$V_NAMES_8BIT フラグおよび FIB$V_NAMES_16BIT フラグの設定によって制御されます。名前が返される形式が，呼び出し元のプログラムで受け入れられる形式でない場合は，実際のファイル指定の代わりに疑似名が返されることがあります。

出力ファイル指定パラメータの場合とは異なり， ASCNAME 属性に含まれているファイル指定の長さは，明示的には返されません。このファイル指定の長さを判断するには，呼び出し元のプログラムは，属性バッファの中で最初に現れるパディング文字を検索しなければなりません。 FIB$V_NAMES_8BIT フラグも FIB$V_NAMES_16BIT フラグも設定されていない場合には，バッファにはスペースがパディングされます (この場合には， ODS-2 形式の名前だけが返されることに注意してください)。 1 つまたは複数のフラグが設定されている場合には，属性バッファには 0 (ゼロ) がパディングされます。

注意

ファイル・システムは，属性バッファについては，長さの最小値を制限しません。ファイル指定が属性バッファより長い場合，返される値はエラーを通知または警告せずに長さが切り捨てられます。

これとは対照的に，ファイル・システムは属性バッファについては，長さの最大値を制限します。この最大値よりも大きいバッファを指定すると， BADPARAM **エラーが返されます。**

ATR$C_ASCNAME **属性の書き込み**

ASCNAME 属性は， FIB$V_NAMES_8BIT フラグおよび FIB$V_NAMES_16BIT フラグがクリアされている場合に限って， ODS-2 ボリュームまたは ODS-5 ボリューム上のファイルについてのみ書き込むことができます。

この属性を書き込む機能は，この機能を持つ既存のアプリケーションとの互換性を保つためにだけ提供されています。新しいプログラムや変更されたプログラムは，この属性を書き込むべきではありません。この属性の値を変更すると，ファイルを永久的に削除できなくなることがあります。

属性を書き込むことが可能な場合には，属性バッファの内容 (最大で 252 バイトまで) は，ファイル・ヘッダの中のファイル名フィールドにコピーされます。 ODS-5 ヘッダの場合，形式は ODS-2 に設定され，ファイル名の長さは，最初のスペース文字のオフセットに設定されます。この値は， 252 バイトまたは用意されたバッファの長さのいずれか小さい方の値になります。

ATR$C_FILE_SPEC

FILE_SPEC 属性は，物理ファイル指定を以下の形式で返す，読み込み専用の属性です。

```
DDnn:[DIR1.DIR2_DIRn]name.type;1
```

返されるファイル名はファイル名に含まれているため，ディレクトリの中のファイル名とは異なっていることがあります。パスにあるいずれかのディレクトリのファイル・ヘッダを読み込むときにエラーが発生すると，このファイル指定は，不完全になることがあります。

ODS-5 ボリューム上のファイルの場合，パスには 3 つの名前形式のうちいずれかの形式のファイル名が含まれていることがあります。このため，たとえばディレクトリ名の中にピリオドが含まれているため，返されるパス指定が曖昧になるというような問題が多数発生する可能性があります。このような問題が発生することを避けるために，ファイル・システムは， RMS によって提供されるサービスを利用して，パスのファイル指定やコンポーネントをエスケープ形式に変換します。[5]

パスのエスケープ形式がその属性のバッファで対応できる長さより長すぎる場合には，パスの中の 1 つまたは複数のディレクトリが，最も右のディレクトリの DID に置き換えられることがあります。この処理は， RMS によって実行される処理とまったく同じもので，第 B.2 節で詳しく説明しています。 DID による短縮を実行した後でも，ファイル指定がバッファで対応できる長さより長すぎる場合，そのファイル名は切り捨てられます。ユーザ・バッファに返されるファイル指定文字列には， 2 バイトのカウント接頭辞が付いています。このカウント接頭辞には，切り捨てられなかったファイル指定のバイト数が含まれています。このカウントがユーザ・バッファのサイズ (カウントが含まれているバイト数から 2 バイトを減じた値) より大きい場合，ユーザは返されたファイル指定が切り捨てられたと判断することができます。

ATR$C_FILNAM，ATR$C_FILTYP，および ATR$C_FILVER
これら 3 つの属性のうち最初の 2 つを使用すると，Radix–50によるエンコードを使用してファイル名およびファイル・タイプを読み書きすることができます。このエンコード方式を使用すると， 3 つの文字を 16 ビットの 1 ワードにパックすることができます。Radix–50の名前に使用することができるのは， ODS-2 形式セットでは 38 文字だけですが，ダッシュ (-) とアンダスコア (_) は例外で，使用することはできません。

Radix–50によってエンコードされるファイル指定のコンポーネントの長さの最大値は，以下のとおりです。

- ファイル名: 15 文字 (10 バイト)
- ファイル・タイプ: 6 文字 (4 バイト)

このような文字数および長さの制約があるため，Radix–50によるエンコードで表現できるのは，有効な ODS-2 形式のファイル名のサブセットだけになります。

ヘッダの中の既存のファイル名の形式が ODS-2 の場合，ファイル・システムは，これら 3 つの属性だけを読み書きしようとします。それ以外の場合は， NORAD50 エラーが返されます。既存のファイル名が ODS-2 形式である場合でも，Radix–50によるエンコードに対応していないか，またはRadix–50ファイル名の長さの制限に対応していない場合は， BADFILENAME エラーが返されます。

[5] ODS-5 ボリューム上のファイルに複合アーキテクチャの OpenVMS システムの VAX システムからアクセスすると，エスケープ形式は返されません。 ODS-2 ファイル形式または ISO Latin-1 ファイル形式の場合，ファイル・ヘッダに格納されている名前が返されます。 UCS-2 ファイル形式の場合，疑似名の後に識別子が括弧で囲まれて続いたものが返されます。次にファイル形式の例を示します。

```
DKA100:[ABC]\pUNICODE\.???  (10095,5,0)
```

ATR$C_FILVER 属性を使用すると，ファイル・ヘッダの中のファイル・バージョン番号を 2 バイトの整数として読み書きすることができます。既存のファイル名をRadix–50ファイル名に変換する必要がある場合は，これらの制約はこの属性にも適用されます。

B.4 プログラミング・ユーティリティの変更点

この後の項では， OpenVMS プログラミング・ユーティリティおよびそのルーチンに固有の， Extended File Specifications をサポートするための変更点について説明します。

B.4.1 ファイル定義言語 (FDL) ルーチン

Alpha システム対応の OpenVMS バージョン 7.2 では，ファイル定義言語 (FDL) ルーチンが Extended File Specifications をサポートするように機能強化されています。以下のルーチンのflags引数には，新しいフラグ FDL$V_LONG_NAMES が追加されました。

FDL$CREATE
FDL$GENERATE
FDL$PARSE
FDL$RELEASE

この後の項では，それぞれの FDL ルーチンでの FDL$V_LONG_NAMES フラグの機能について説明します。

B.4.1.1 FDL$CREATE ルーチン (Alpha システムのみ)

flags引数に，以下の新しいフラグが追加されました。

フラグ	機能
FDL$V_LONG_NAMES	長い名前アクセス・ブロック (NAML) の長い結果名を使用して RESULT_NAME を返す。省略時の設定では，RESULT_NAME は名前アクセス・ブロック (NAM) の短いフィールドから返されるため，生成された指定が含まれていることがある。 このフラグは， OpenVMS Alpha システムでのみ有効である。

B.4.1.2 FDL$GENERATE Routine (Alpha システムのみ)

flags引数に，以下の新しいフラグが追加されました。

フラグ	機能
FDL$V_LONG_NAMES	長い名前アクセス・ブロック (NAML) の長い結果名を使用して FDL_FILE_RESNAM を返す。省略時の設定では，FDL_FILE_RESNAM は名前アクセス・ブロック (NAM) の短いフィールドから返されるため，生成された指定が含まれていることがある。 このフラグは， OpenVMS Alpha システムでのみ有効である。

B.4.1.3 FDL$PARSE ルーチン (Alpha システムのみ)

flags引数に，以下の新しいフラグが追加されました。

フラグ	機能
FDL$V_LONG_NAMES	返される RMS ファイル・アクセス・ブロック (FAB) にリンクされた長い名前アクセス・ブロックを割り当てて返す。NAML ブロックの長いファイル名フィールドが使用されるように， NAML ブロックと FAB ブロックには適切な値が設定される。 省略時の設定では，名前ブロックは割り当てられず， FAB のファイル名フィールドが使用される。 FDL$V_LONG_NAMES フラグが設定されている場合には， NAML ブロックに割り当てられたメモリが適切に割り当て解除されるように， FDL$RELEASE ルーチンのflags引数にも FDL$V_LONG_NAMES ビットが設定されていなければならない。 このフラグは， OpenVMS Alpha システムでのみ有効である。

B.4.1.4 FDL$RELEASE ルーチン (Alpha システムのみ)

flags引数に，以下の新しいフラグが追加されました。

フラグ	機能
FDL$V_LONG_NAMES	FDL$PARSE ルーチンによって作成された長い名前アクセス・ブロック (NAML) で使用する仮想メモリがあれば，これを割り当て解除する。 このフラグは， OpenVMS Alpha システムでのみ有効である

B.5 実行時ライブラリの変更点

拡張ファイル名を使用できるようにするために，オプションで NAM ブロックではなく NAML ブロックを受け付けたり返すことができるように， LIB$実行時ライブラリのいくつかのルーチンが変更されています。以下のルーチンが変更されました。

- LIB$DELETE_FILE
- LIB$FILE_SCAN

- LIB$FIND_FILE
- LIB$RENAME_FILE
- LIB$FID_TO_NAME

B.5.1 LIB$DELETE_FILE

LIB$DELETE_FILE の形式は，以下のようになりました。

```
LIB$DELETE_FILE     filespec [,default-filespec] [,related-filespec]
                    [,user-success-procedure] [,user-error-procedure]
                    [,user-confirm-procedure] [,user-specified-argument]
                    [,resultant-name] [,file-scan-context] [,flags])
```

flags引数は新しい引数で，形式は以下のようになっています。

```
 flags
OpenVMS usage:   mask_longword
type:            longword (unsigned)
access:          read only
mechanism:        by reference
```

ユーザ・フラグ。flags引数は，ユーザ・フラグが含まれている符号なしロング・ワードのアドレスです。

次の表に，フラグ・ビットおよびそれらに対応するシンボルを示します。

ビット	シンボル	説明
0		Compaq で使用するために予約済み。
1		Compaq で使用するために予約済み。
2	LIB$M_FIL_LONG_NAMES	(Alpha システムのみ) これが設定されている場合， LIB$DELETE_FILE は，最大長が NAML$C_MAXRSS のファイル名を処理することができる。これがクリアされている場合， LIB$DELETE_FILE は，最大長が 255 バイト (省略時の値) のファイル指定を処理することができる。

Alpha システムでは，LIB$DELETE_FILEへの呼び出しで引数user-confirm-procedureを指定し，LIB$M_FIL_LONG_NAMESフラグが設定されている場合，confirm-procedure ルーチンのfab引数によって参照される FAB は， NAM ブロックではなく NAML ブロックを参照します。 NAML ブロックでは，最大長が NAML$C_MAXRSS の長いファイル名がサポートされています。 NAML ブロックの詳細については，『OpenVMS Record Management Services Reference Manual』を参照してください。

LIB$DELETE_FILE には，返される新しい条件値が 1 つあります。 LIB$INVARG は，flags引数に指定されていないビットが設定されたことを示します。

B.5.2 LIB$FILE_SCAN

LIB$FILE_SCAN のfab引数は， NAM ブロックまたは NAML ブロックのいずれかを参照できるようになりました。

B.5.3 LIB$FIND_FILE

LIB$FIND_FILE のflags引数には，以下の新しいビットが追加されています。

ビット	シンボル	説明
2	LIB$M_FIL_LONG_NAMES	(Alpha システムのみ) これが設定されている場合， LIB$FIND_FILE は，最大長が NAML$C_MAXRSS のファイル名を処理することができる。これがクリアされている場合， LIB$FIND_FILE は，最大長が 255 バイト (省略時の値) のファイル指定を処理することができる。

Alpha システムでは，flags引数に LIB$M_FIL_LONG_NAMES フラグが設定されている場合に限って， 255 バイトより長いファイル指定がサポートされます。このフラグが設定されている場合， NAML ブロック (NAM ブロックではない) がコンテキストの一部になり，ファイル指定の長さを最大で NAML$C_MAXRSS にすることができます。 NAML ブロックの詳細については，『OpenVMS Record Management Services Reference Manual』を参照してください。

B.5.4 LIB$RENAME_FILE

LIB$RENAME_FILE のflags引数には，以下の新しいビットが追加されています。

ビット	シンボル	説明
2	LIB$M_FIL_LONG_NAMES	(Alpha システムのみ) 長さが 255 バイトを超えるファイル指定を受け付けるかどうかを制御する。 これが設定されている場合， LIB$RENAME_FILE は，最大長が NAML$C_MAXRSS のファイル名を処理することができる。これがクリアされている場合， LIB$RENAME_FILE は，最大長が 255 バイト (省略時の値) のファイル指定を処理することができる。

Alpha システムでは，LIB$RENAME_FILEへの呼び出しでuser-confirm-procedureを指定し，LIB$M_FIL_LONG_NAMESフラグが設定されている場合， confirm-procedure ルーチンのfab引数によって参照される FAB は， NAM ブロックではなく NAML ブロックを参照します。 NAML ブロックでは，最大長が NAML$C_MAXRSS の長いファイル名がサポートされています。 NAML ブロックの詳細につい

ては，『OpenVMS Record Management Services Reference Manual』を参照してください。

B.5.5 LIB$FID_TO_NAME

ファイル指定の長さがfilespecバッファで対応できる長さを超えると，パス内のディレクトリが DID による短縮に置き換えられることがあります (第 3.2.2.1 項を参照)。DID による短縮が実行された後でもファイル指定の長さがバッファで対応できる長さを超えている場合は，そのファイル指定の長さが切り捨てられ，以前のバージョンの場合と同様に，代わりの正常終了状態として LIB$_STRTRU が返されます。動的記述子の場合には，最大値を使用することにより，最大で 4095 バイトまでの文字列を返すことができます。

C
文字セット

DEC Multinational 文字セット (MCS) は， Digital Equipment Corporation で作成され使用された 00 ～ FF の 16 進数で表現される文字の定義から成ります。 DEC MCS は， 7 ビットの (00 ～ 7F の 16 進数値で表現される) ASCII 文字セットと， 80 ～ FF の 16 進数で表現される 8 ビット文字のセットの 2 つに分類されます。 DEC MCS は， Digital Equipment Corporation で作成され販売されたソフトウェアのほとんどのユーザにとって使い慣れている文字セットです。

Unicode Standard Character Set (UCS-2) は， The Unicode Consortium によって定義された， 0000 ～ FFFF の値で表現される 16 ビット文字のセットです。

ISO Latin-1 文字セットは， 00 ～ FF の 16 進数で表現される 8 ビット文字です。 ISO Latin-1 文字セットの定義は， 80 ～ FF の 16 進数値から成る DEC MCS の定義とは少し異なります。

表 C–1 には， DEC Multinational 文字セット (MCS) が示されています。表 C–1 は， 2 種類の文字セットでの文字の違いを示し，図 C–1 は，異なっている文字を示しています。

Unicode (UCS-2) 文字セットの詳細については， The Unicode Consortium から発行されている『The Unicode Standard』を参照してください。

表 C–1 DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		ASCII 制御文字[1]
00	NUL	空文字
01	SOH	ヘッダの始点 (Ctrl/A)
02	STX	テキストの始点 (Ctrl/B)
03	ETX	テキストの終端 (Ctrl/C)
04	EOT	転送の終端 (Ctrl/D)
05	ENQ	問い合わせ (Ctrl/E)

[1]ALTMODE 文字および DELETE 文字 (10 進数の 125， 126，および 127) も，制御文字です。

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		ASCII 制御文字[1]
06	ACK	肯定応答 (Ctrl/F)
07	BEL	ベル (Ctrl/G)
08	BS	バックスペース (Ctrl/H)
09	HT	水平タブ (Ctrl/I)
0A	LF	ライン・フィード (Ctrl/J)
0B	VT	垂直タブ (Ctrl/K)
0C	FF	フォーム・フィード (Ctrl/L)
0D	CR	キャリッジ・リターン (Ctrl/M)
0E	SO	シフト・アウト (Ctrl/N)
0F	SI	シフト・イン (Ctrl/O)
10	DLE	データ・リンク・エスケープ (Ctrl/P)
11	DC1	デバイス制御 1 (Ctrl/Q)
12	DC2	デバイス制御 2 (Ctrl/R)
13	DC3	デバイス制御 3 (Ctrl/S)
14	DC4	デバイス制御 4 (Ctrl/T)
15	NAK	否定応答 (Ctrl/U)
16	SYN	同期アイドル (Ctrl/V)
17	ETB	転送ブロックの終端 (Ctrl/W)
18	CAN	取り消し (Ctrl/X)
19	EM	媒体の終端 (Ctrl/Y)
1A	SUB	置換 (Ctrl/Z)
1B	ESC	エスケープ
1C	FS	ファイル区切り文字
1D	GS	グループ区切り文字
1E	RS	レコード区切り文字
1F	US	ユニット区切り文字
		ASCII 特殊文字および数値文字
20	SP	スペース
21	!	感嘆符
22	"	引用符 (二重引用符)
23	#	ポンド記号
24	$	ドル記号
25	%	パーセント記号
26	&	アンパサンド

[1]ALTMODE 文字および DELETE 文字 (10 進数の 125, 126, および 127) も，制御文字です。

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
	ASCII 特殊文字および数値文字	
27	′	アポストロフィ (一重引用符)
28	(	左括弧
29	)	右括弧
2A	*	アスタリスク
2B	+	正符号
2C	,	コンマ
2D	–	ハイフンまたは負符号
2E	.	ピリオドまたは小数点
2F	/	スラッシュ
30	0	ゼロ (0)
31	1	1
32	2	2
33	3	3
34	4	4
35	5	5
36	6	6
37	7	7
38	8	8
39	9	9
3A	:	コロン
3B	;	セミコロン
3C	<	不等号 (左辺は右辺より小さい)
3D	=	等号
3E	>	不等号 (左辺は右辺より大きい)
3F	?	疑問符
	ASCII 英文字	
40	@	アットマーク
41	A	大文字の A
42	B	大文字の B
43	C	大文字の C
44	D	大文字の D
45	E	大文字の E
46	F	大文字の F
47	G	大文字の G

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		ASCII 英文字
48	H	大文字の H
49	I	大文字の I
4A	J	大文字の J
4B	K	大文字の K
4C	L	大文字の L
4D	M	大文字の M
4E	N	大文字の N
4F	O	大文字の O
50	P	大文字の P
51	Q	大文字の Q
52	R	大文字の R
53	S	大文字の S
54	T	大文字の T
55	U	大文字の U
56	V	大文字の V
57	W	大文字の W
58	X	大文字の X
59	Y	大文字の Y
5A	Z	大文字の Z
5B	[	左大括弧
5C	\	バックスラッシュ
5D	]	右大括弧
5E	^	サーカンフレックス
5F	_	アンダスコア
60	‘	低アクセント
61	a	小文字の a
62	b	小文字の b
63	c	小文字の c
64	d	小文字の d
65	e	小文字の e
66	f	小文字の f
67	g	小文字の g
68	h	小文字の h
69	i	小文字の i
6A	j	小文字の j

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		ASCII 英文字
6B	k	小文字の k
6C	l	小文字の l
6D	m	小文字の m
6E	n	小文字の n
6F	o	小文字の o
70	p	小文字の p
71	q	小文字の q
72	r	小文字の r
73	s	小文字の s
74	t	小文字の t
75	u	小文字の u
76	v	小文字の v
77	w	小文字の w
78	x	小文字の x
79	y	小文字の y
7A	z	小文字の z
7B	{	左中括弧
7C	|	縦線
7D	}	右中括弧 (ALTMODE)
7E	~	チルダ (ALTMODE)
7F	DEL	削除 (DELETE)
		制御文字
80		[予約領域]
81		[予約領域]
82		[予約領域]
83		[予約領域]
84	IND	索引
85	NEL	次の行
86	SSA	選択領域の始点
87	ESA	選択領域の終端
88	HTS	水平タブの設定
89	HTJ	行そろえを使用した水平タブの設定
8A	VTS	垂直タブの設定
8B	PLD	行単位での部分的な下方向への移動

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		制御文字
8C	PLU	行単位での部分的な上方向への移動
8D	RI	逆方向の索引
8E	SS2	シングルシフト 2
8F	SS3	シングルシフト 3
90	DCS	デバイス制御文字列
91	PU1	プライベート使用 1
92	PU2	プライベート使用 2
93	STS	転送状態の設定
94	CCH	文字の取り消し
95	MW	メッセージの待機
96	SPA	保護領域の始点
97	EPA	保護領域の終端
98		[予約領域]
99		[予約領域]
9A		[予約領域]
9B	CSI	制御シーケンス・イントロデューサ
9C	ST	文字列終了文字列
9D	OSC	オペレーティング・システム・コマンド
9E	PM	プライバシー・メッセージ
9F	APC	アプリケーション
		その他の文字
A0		[予約領域][2]
A1	¡	逆感嘆符
A2	¢	セント記号
A3	£	貨幣のポンド記号
A4		[予約領域][2]
A5	¥	円記号
A6		[予約領域][2]
A7	§	セクション記号
A8	¤	一般貨幣記号[2]
A9	©	著作権記号
AA	ª	女性形序数指示子
AB	«	左角引用符
AC		[予約領域][2]

[2]ISO Latin-1 では別の文字です。図 C–1 を参照してください。

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		その他の文字
AD		[予約領域][2]
AE		[予約領域][2]
AF		[予約領域][2]
B0	°	温度記号
B1	±	正負符号
B2	²	スーパースクリプト 2
B3	³	スーパースクリプト 3
B4		[予約領域][2]
B5	µ	ミクロ記号
B6	¶	段落記号，段落標
B7	·	中黒
B8		[予約領域][2]
B9	¹	スーパースクリプト 1
BA	º	男性形序数指示子
BB	»	右角引用符
BC	¼	小数部四分の一
BD	½	小数部二分の一
BE		[予約領域][2]
BF	¿	逆疑問符
C0	À	低アクセント付きの大文字の A
C1	Á	鋭アクセント付きの大文字の A
C2	Â	サーカンフレックス付きの大文字の A
C3	Ã	チルダ付きの大文字の A
C4	Ä	ウムラウト付きの大文字の A (分音符号)
C5	Å	リング付きの大文字の A
C6	Æ	大文字の AE 二重母音
C7	Ç	セディーユ付きの大文字の C
C8	È	低アクセント付きの大文字の E
C9	É	鋭アクセント付きの大文字の E
CA	Ê	サーカンフレックス付きの大文字の E
CB	Ë	ウムラウト付きの大文字の E (分音符号)
CC	Ì	低アクセント付きの大文字の I
CD	Í	鋭アクセント付きの大文字の I
CE	Î	サーカンフレックス付きの大文字の I
CF	Ï	ウムラウト付きの大文字の I (分音符号)

[2]ISO Latin-1 では別の文字です。図 C–1 を参照してください。

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		その他の文字
D0		[予約領域][2]
D1	Ñ	チルダ付きの大文字の N
D2	Ò	低アクセント付きの大文字の O
D3	Ó	鋭アクセント付きの大文字の O
D4	Ô	サーカンフレックス付きの大文字の O
D5	Õ	チルダ付きの大文字の O
D6	Ö	ウムラウト付きの大文字の O (分音符号)
D7	Œ	大文字の OE 合字[2]
D8	Ø	スラッシュ付きの大文字の O
D9	Ù	低アクセント付きの大文字の U
DA	Ú	鋭アクセント付きの大文字の U
DB	Û	サーカンフレックス付きの大文字の U
DC	Ü	ウムラウト付きの大文字の U (分音符号)
DD	Ÿ	ウムラウト付きの大文字の Y (分音符号)
DE		[予約領域][2]
DF	ß	ドイツ語の小文字の鋭い s (s ツェット)
E0	à	低アクセント付きの小文字の a
E1	á	鋭アクセント付きの小文字の a
E2	â	サーカンフレックス付きの小文字の a
E3	ã	チルダ付きの小文字の a
E4	ä	ウムラウト付きの小文字の a (分音符号)
E5	å	リング付きの小文字の a
E6	æ	小文字の ae 二重母音
E7	ç	セディーユ付きの小文字の c
E8	è	低アクセント付きの小文字の e
E9	é	鋭アクセント付きの小文字の e
EA	ê	サーカンフレックス付きの小文字の e
EB	ë	ウムラウト付きの小文字の e (分音符号)
EC	ì	低アクセント付きの小文字の i
ED	í	鋭アクセント付きの小文字の i
EE	î	サーカンフレックス付きの小文字の i
EF	ï	ウムラウト付きの小文字の i (分音符号)
F0		[予約領域][2]
F1	ñ	チルダ付きの小文字の n
F2	ò	低アクセント付きの小文字の o

[2]ISO Latin-1 では別の文字です。図 C–1 を参照してください。

(次ページに続く)

表 C–1 (続き) DEC Multinational 文字セット

16 進数のコード	MCS 文字または短縮形	DEC Multinational 文字名
		その他の文字
F3	ó	鋭アクセント付きの小文字の o
F4	ô	サーカンフレックス付きの小文字の o
F5	õ	チルダ付きの小文字の o
F6	ö	ウムラウト付きの小文字の o (分音符号)
F7	œ	小文字の oe 合字[2]
F8	ø	スラッシュ付きの小文字の o
F9	ù	低アクセント付きの小文字の u
FA	ú	低アクセント付きの小文字の u
FB	û	サーカンフレックス付きの小文字の u
FC	ü	ウムラウト付きの小文字の u (分音符号)
FD	ÿ	ウムラウト付きの小文字の y (分音符号)[2]
FE		[予約領域][2]
FF		[予約領域][2]

[2]ISO Latin-1 では別の文字です。図 C–1 を参照してください。

図 C–1 DEC Multinational **文字セットと** ISO Latin-1 **文字セットの違い**

Hex Code	MCS Char or Abbrev.	DEC Multinational Character Name	Isolatin-1 Char	Isolatin-1 Character Name
A0		[reserved]		nonbreaking space
A4		[reserved]	¤	currency sign
A6		[reserved]	¦	broken vertical bar
A8	¤	currency sign	¨	spacing diaeresis
AC		[reserved]	¬	not sign
AD		[reserved]		soft hyphen
AE		[reserved]	®	registered trademark sign
AF		[reserved]	¯	spacing macron
B4		[reserved]	´	spacing acute
B8		[reserved]	¸	spacing cedilla
BE		[reserved]	¾	fraction three quarters
D0		[reserved]	Ð	Latin capital letter eth
D7	Œ	uppercase OE ligature	×	multiplication sign
DE		[reserved]	Þ	Latin capital letter thorn
F0		[reserved]	ð	Latin small letter eth
F7	œ	lowercase oe ligature	÷	division sign
FD	ÿ	lowercase y with umlaut, (diaeresis)	ý	Latin small letter y acute
FE		[reserved]	þ	Latin small letter thorn
FF		[reserved]	ÿ	Latin small letter y diaeresis

VM-0128A-AI

索引

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

ア

エ

オ

カ

キ

コ

サ

シ

セ

タ

チ

テ

ト

ハ

フ

ヘ

モ

OpenVMS Extended File Specifications **の手引き**

1999 年 4 月　発行

コンパックコンピュータ株式会社

〒 140-8641 東京都品川区東品川 2-2-24 天王洲セントラルタワー

電話　(03)5463-6600 (大代表)

AA-RFJEA-TE